ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΠΑΙΔΑΓΩΓΙΚΟ ΙΝΣΤΙΤΟΥΤΟ

ΤΕΧΝΟΛΟΓΙΑ ΥΠΟΛΟΓΙΣΤΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ & ΛΕΙΤΟΥΡΓΙΚΑ ΣΥΣΤΗΜΑΤΑ

ΑΘΗΝΑ 1999

ΦΙΛΙΠΠΟΣ ΜΑΚΡΥΤΗΣ
Σ. ΚΩΝΣΤΑΝΤΙΝΟΣ
Γεωργάκης Κωνσταντίνος

Κεφάλαιο

Κατηγορίες Υπολογιστικών Συστημάτων

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει την εξέλιξη των υπολογιστικών συστημάτων, τις κατηγορίες τους και τον ιεραρχικό τρόπο οργάνωσής τους· επίσης να σε ενημερώσει για τα ολοκληρωμένα κυκλώματα.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις από τι αποτελείται ένα υπολογιστικό σύστημα και πώς είναι οργανωμένο.
- Να αναφέρεις τις τεχνολογίες των ολοκληρωμένων κυκλωμάτων.
- Να απαριθμείς τις γενιές και τις κατηγορίες των υπολογιστικών συστημάτων.

Μαθήματα

1.1 Οργάνωση και Εξέλιξη των Υπολογιστικών Συστημάτων

Μάθημα 1.1 Οργάνωση και Εξέλιξη των Υπολογιστικών Συστημάτων

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τα τμήματα και τα επίπεδα ιεραρχίας ενός υπολογιστικού συστήματος, την εξέλιξη των ολοκληρωμένων κυκλωμάτων καθώς και των υπολογιστικών συστημάτων.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να αναφέρεις από τι αποτελείται ένα υπολογιστικό σύστημα*
- *Να περιγράφεις πώς γίνεται η ιεραρχική οργάνωση στα υπολογιστικά συστήματα*
- *Να εξηγείς τι είναι ένα ολοκληρωμένο κύκλωμα*
- *Να αναφέρεις τις τεχνολογίες των ολοκληρωμένων κυκλωμάτων*
- *Να απαριθμείς τις γενιές των υπολογιστικών συστημάτων*
- *Να κατονομάζεις τις κατηγορίες στις οποίες χωρίζονται τα υπολογιστικά συστήματα με βάση τις δυνατότητές τους*

Κάθε υπολογιστικό σύστημα αποτελείται από το υλικό (hardware) και το λογισμικό (software).

> Το *υλικό* (hardware) του υπολογιστή είναι το σύνολο των συσκευών που απαρτίζουν το υπολογιστικό σύστημα.

Το υλικό λοιπόν περιλαμβάνει τα ολοκληρωμένα κυκλώματα του υπολογιστή, τους δίσκους, τους εκτυπωτές, την οθόνη κλπ.

> Το *λογισμικό* (software) ορίζεται ως το σύνολο των προγραμμάτων, τα οποία μπορούν να εκτελεσθούν από το υπολογιστικό σύστημα.

Αν και τα προγράμματα μπορεί να είναι αποθηκευμένα σε μέσα όπως οι δίσκοι, εντούτοις το λογισμικό δεν ταυτίζεται με τα μέσα αυτά, αλλά με τις εντολές που είναι αποθηκευμένες στα μέσα και συνιστούν τα προγράμματα.

Το λογισμικό αποτελείται από το *λογισμικό του συστήματος* (system software) και το *λογισμικό των εφαρμογών* (application software).

⇨ Το **λογισμικό του συστήματος** (το μεγαλύτερο μέρος του οποίου είναι το λειτουργικό σύστημα) λειτουργεί ως σύνδεσμος ανάμεσα στο χρήστη και στο υλικό, με σκοπό τη διευκόλυνση του χρήστη και την αποδοτικότερη λειτουργία του υπολογιστικού συστήματος.

⇨ Το **λογισμικό των εφαρμογών** είναι τα προγράμματα που γράφονται για να καλύψουν τις ειδικές ανάγκες των χρηστών και καθορίζουν τον τρόπο που θα χρησιμοποιηθεί ο υπολογιστής για την επίλυση συγκεκριμένων υπολογιστικών προβλημάτων.

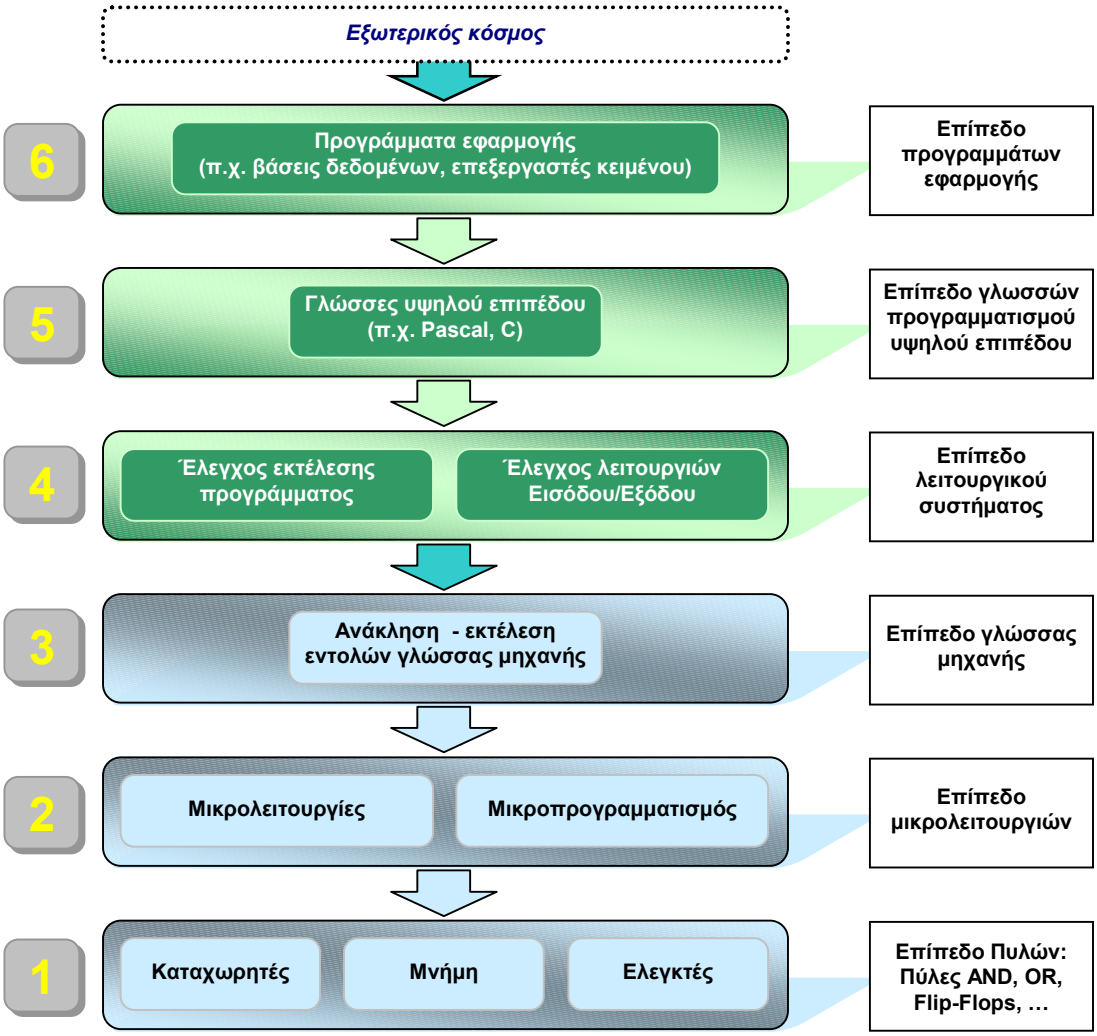
Ιεραρχική οργάνωση των υπολογιστικών συστημάτων

Τα υπολογιστικά συστήματα χαρακτηρίζονται από πολυπλοκότητα και ιεραρχική οργάνωση.

Η **πολυπλοκότητα** των υπολογιστικών συστημάτων οφείλεται στο ότι αποτελούνται από πολλά μέρη, τα οποία αλληλεπιδρούν δυναμικά μεταξύ τους.

Η **ιεραρχική οργάνωση** των υπολογιστικών συστημάτων είναι απαραίτητη εξαιτίας της πολυπλοκότητας αυτής. Τα τμήματα του υπολογιστικού συστήματος κατανέμονται σε οργανωτικά «επίπεδα». Η λειτουργία κάθε επιπέδου στηρίζεται στη λειτουργία των χαμηλότερων επιπέδων, και με τη σειρά του κάθε επίπεδο βοηθά στη λειτουργία των ανωτέρων του επιπέδων. Κάθε ένα από τα επίπεδα μπορεί να σχεδιαστεί, να υλοποιηθεί, να μελετηθεί και να κατανοηθεί ανεξάρτητα από τα υπόλοιπα.

Στο σχήμα βλέπουμε ένα τρόπο ανάλυσης ενός υπολογιστικού συστήματος σε επίπεδα ιεραρχίας.



Επίπεδα ιεραρχίας

Τα επίπεδα 1, 2 και 3 που βλέπουμε στο προηγούμενο σχήμα αφορούν το υλικό τού υπολογιστικού συστήματος.

1 Το **επίπεδο 1**, αυτό των πυλών (gates), είναι το χαμηλότερο επίπεδο στην ιεραρχία. Στο επίπεδο αυτό οι βασικές λειτουργίες επιτελούνται από στοιχειώδη λογικά κυκλώματα όπως καταχωρητές, μνήμες, ελεγκτές λογικών κυκλωμάτων κλπ.
Στο επίπεδο 1 υπάρχουν οι βασικές δομικές μονάδες από τις οποίες συντίθεται ένα υπολογιστικό σύστημα και ονομάζονται *λογικά κυκλώματα* (logical circuits), δηλαδή οι λογικές πύλες, τα flip-flops, οι καταχωρητές κ.ά.

2 Το **επίπεδο 2** αφορά τις στοιχειώδεις *μικρολειτουργίες* (microoperations) που επιτελούνται στο υπολογιστικό σύστημα (π.χ. μεταφορά πληροφοριών). Για την υλοποίηση των μικρολειτουργιών απαιτείται η ύπαρξη των βασικών κυκλωμάτων του επιπέδου 1.

3 Το **επίπεδο 3** αφορά τη διαδικασία της ανάκλησης από τη μνήμη και της εκτέλεσης εντολών γλώσσας μηχανής. Για τη διαδικασία αυτή χρησιμοποιούνται οι μικρολειτουργίες του επιπέδου 2.

Τα **επίπεδα 4, 5 και 6** περιλαμβάνουν το λογισμικό, δηλαδή το λειτουργικό σύστημα, τις γλώσσες υψηλού επιπέδου και τα προγράμματα εφαρμογής, αντίστοιχα.

Κάθε ένα επίπεδο μπορεί να αναλυθεί σε άλλα, λεπτομερέστερα, επίπεδα ιεραρχίας.

Στο βιβλίο αυτό θα ασχοληθούμε κυρίως με τα επίπεδα 3 και 4.

Ολοκληρωμένα κυκλώματα

Η ραγδαία τεχνολογική εξέλιξη στον τομέα των ηλεκτρονικών έχει κάνει δυνατή την κατασκευή πολύπλοκων ηλεκτρονικών κυκλωμάτων σε ένα μικρό τεμάχιο ημιαγωγού (π.χ. πυριτίου) διαστάσεων μερικών τετραγωνικών χιλιοστών, στο οποίο με κατάλληλες τεχνικές σχηματίζονται τα διάφορα στοιχεία, όπως κρυσταλλοδίοδοι, κρυσταλλοτρίοδοι, πυκνωτές, καθώς και οι συνδέσεις τους. Τα στοιχεία αυτά σχηματίζουν ένα *ολοκληρωμένο κύκλωμα* (Integrated Circuit, IC). Η όλη κατασκευή τοποθετείται σε μεταλλική ή πλαστική συσκευασία που αποτελεί τη λεγόμενη *ψηφίδα* (chip). Η ψηφίδα επικοινωνεί με άλλα κυκλώματα με εξωτερικούς ακροδέκτες, τις ακίδες. Το μέγεθος μίας ψηφίδας είναι μερικά τετραγωνικά εκατοστά, μαζί με το πλαστικό της περίβλημα και τους ακροδέκτες.

Μια από τις περισσότερο χρησιμοποιούμενες οικογένειες ολοκληρωμένων κυκλωμάρων είναι η TTL (Transistor Transistor Logic, Κυκλώματα Τρανζίστορ - Τρανζίστορ) ενώ για κυκλώματα υψηλών ταχυτήτων χρησιμοποιείται η οικογένεια ECL (Emitter Coupled Logic, Κυκλώματα Συνδεδεμένου Εκπομπού). Τα κυκλώματα που είναι υλοποιημένα με CMOS (Complementary Metal Oxide Semiconductor, Κυκλώματα MOS με Συμπληρωματικά Στοιχεία) έχουν μικρή κατανάλωση ισχύος.

Τα στοιχεία των ηλεκτρονικών κυκλωμάτων που περιέχονται στις ψηφίδες συνδυάζονται για να σχηματίσουν *πύλες* (gates). Μία πύλη δέχεται σαν είσοδο δύο ή περισσότερα ηλεκτρικά σήματα τα οποία κωδικοποιούν bits (π.χ. τα 5 Volt παριστάνουν το 1 και τα 0 Volt παριστάνουν το 0) και παράγει στην έξοδο ένα σήμα το οποίο είναι το αποτέλεσμα μίας «πράξης» μεταξύ των εισόδων της.

Η πύλη AND («ΚΑΙ») δίνει στην έξοδό της 1 αν όλα τα σήματα εισόδου της έχουν την τιμή 1, αλλιώς δίνει στην έξοδο 0.

Μια άλλη διάκριση που γίνεται στα ολοκληρωμένα κυκλώματα, είναι ανάλογα με τον αριθμό των δομικών πυλών που περιέχει κάθε ψηφίδα. Έτσι, διακρίνονται τα κυκλώματα:

1. *Μικρής Κλίμακας Ολοκλήρωσης* (Small Scale Integration, SSI). Αυτά περιέχουν λίγες (5-10) μεμονωμένες πύλες.
2. *Μέσης Κλίμακας Ολοκλήρωσης* (Medium Scale Integration, MSI). Αυτά περιέχουν 10-100 πύλες κατάλληλα συνδεδεμένες, ώστε να σχηματίζουν ένα ψηφιακό κύκλωμα (όπως π.χ. έναν καταχωρητή ή μετρητή).
3. *Μεγάλης Κλίμακας Ολοκλήρωσης* (Large Scale Integration, LSI), τα οποία περιέχουν περισσότερες από 100 πύλες μέχρι μερικές χιλιάδες πύλες (όπως οι απλοί μικροεπεξεργαστές).
4. *Πολύ Μεγάλης Κλίμακας Ολοκλήρωσης* (Very Large Scale Integration, VLSI) που περιέχουν κυκλώματα τα οποία σχηματίζονται από πολλά εκατομμύρια πύλες (όπως οι σύγχρονοι μικροεπεξεργαστές).

Όλα τα ηλεκτρονικά κυκλώματα κατασκευάζονται με ολοκληρωμένα κυκλώματα, αφού τα τελευταία προσφέρουν πολλά πλεονεκτήματα: έχουν μικρό μέγεθος, υψηλή ταχύτητα λειτουργίας, μικρό κόστος, μεγάλη αξιοπιστία, μικρή κατανάλωση ενέργειας και δίνουν μεγάλη ευελιξία στη σχεδίαση και κατασκευή ηλεκτρονικών κυκλωμάτων για υπολογιστές.

Ιστορική εξέλιξη των υπολογιστικών συστημάτων

Τις θεωρητικές βάσεις των υπολογιστών έθεσε πρώτος ο μαθηματικός J. Von Neumann (1945). Προς τιμή του, μία μεγάλη κατηγορία υπολογιστών ονομάζονται *υπολογιστές τύπου Von Neumann* και η λειτουργία τους στηρίζεται στις έννοιες του *αποθηκευμένου προγράμματος* (stored program) και του *μετρητή προγράμματος* (program counter), με τη βοήθεια του οποίου προσδιορίζεται η εκτέλεση του προγράμματος. Τις έννοιες αυτές είχε εμπνευστεί πρώτος ο Charles Babbage τον περασμένο αιώνα.

1 Συστήματα πρώτης γενιάς

Τα συστήματα της *πρώτης γενιάς* (1946-1953) χρησιμοποιούσαν ως βασικές δομικές μονάδες ηλεκτρονικές λυχνίες. Ο πρώτος ηλεκτρονικός υπολογιστής ήταν ο ENIAC (Electronic Numerical Integrator And Computer), που κατασκευάστηκε το 1946 στο Πανεπιστήμιο της Πενσυλβάνιας. Αντιπροσωπευτικός υπολογιστής αυτής της γενιάς είναι ο IBM 701. Οι υπολογιστές της πρώτης γενιάς προγραμματίζονταν απευθείας σε γλώσσα μηχανής. Το γεγονός αυτό, σε συνδυασμό με το υψηλό τους κόστος και τη χαμηλή τους ταχύτητα, έθετε σοβαρούς περιορισμούς στην ευρεία χρήση τους.

2 Συστήματα δεύτερης γενιάς

Τα συστήματα *δεύτερης γενιάς* (1952-1963) χρησιμοποιούσαν ως βασικές δομικές μονάδες κρυσταλλοτριόδους (Transistors). Ο υπολογιστής TRADIC που κατασκευάστηκε το 1954 στα εργαστήρια της Bell, ήταν ο πρώτος υπολογιστής αυτής της γενιάς. Τα τυπωμένα κυκλώματα, οι μνήμες μαγνητικών δακτυλίων, οι συμβολικές γλώσσες, οι γλώσσες υψηλού επιπέδου όπως η FORTRAN, η ALGOL και η COBOL, τα λειτουργικά συστήματα ομαδικής επεξεργασίας κλπ. ήταν τα κύρια χαρακτηριστικά αυτής της γενιάς.

3 Συστήματα τρίτης γενιάς

Στα συστήματα *τρίτης γενιάς* (1962-1975) χρησιμοποιήθηκαν ως βασικές δομικές μονάδες τα ολοκληρωμένα κυκλώματα μικρής κλίμακας ολοκλήρωσης (SSI) και μέσης κλίμακας ολοκλήρωσης (MSI). Μνήμες ημαγωγών, εικονικές μνήμες (βλ. Κεφ. 4), γλώσσες προγραμματισμού υψηλού επιπέδου με «έξυπνους» μεταφραστές, λειτουργικά συστήματα πολυπρογραμματισμού και καταμερισμού χρόνου κλπ. ήταν τα κύρια χαρακτηριστικά αυτής της γενιάς. Οι CDC-7600, IBM 360/91, Illiac IV κλπ. είναι

χαρακτηριστικοί τύποι υπολογιστών της γενιάς αυτής. Την ίδια εποχή εμφανίζονται και οι μίνι-υπολογιστές.

4 Συστήματα τέταρτης γενιάς

Τα συστήματα *τέταρτης γενιάς* (1972-σήμερα) χρησιμοποιούν ως βασικές δομικές μονάδες ολοκληρωμένα κυκλώματα μεγάλης και πολύ μεγάλης κλίμακας ολοκλήρωσης (LSI και VLSI) . Τα χαρακτηριστικά των συστημάτων της τρίτης γενιάς έχουν βελτιωθεί και έχουν χρησιμοποιηθεί αρχιτεκτονικές αγωγού, πολυεπεξεργασίας, μητρώου κλπ. για την κατασκευή υπερυπολογιστών (Supercomputers), όπως ο Cray T3-E, ο MPP κλπ. Την ίδια εποχή εμφανίζονται και οι μικροϋπολογιστές.

5 Συστήματα πέμπτης γενιάς

Τα συστήματα *πέμπτης γενιάς*, τα οποία δεν έχουν διαδοθεί ευρέως, χρησιμοποιούν ολοκληρωμένα κυκλώματα πολύ μεγάλης κλίμακας ολοκλήρωσης (VLSI) και έχουν δύο βασικούς στόχους. Ο πρώτος είναι η επίτευξη στο μέγιστο δυνατό βαθμό της παράλληλης επεξεργασίας (για την αύξηση της ταχύτητας επεξεργασίας). Ο δεύτερος είναι η ανάπτυξη «έξυπνων» υπολογιστικών συστημάτων, με την ενσωμάτωση τεχνικών που χρησιμοποιούνται στον κλάδο της τεχνητής νοημοσύνης.

Κατηγορίες υπολογιστικών συστημάτων

Με βάση το μέγεθος, την ταχύτητα και την τιμή τους, τα υπολογιστικά συστήματα μπορούν να κατηγοριοποιηθούν ως εξής:

- Οι *προσωπικοί υπολογιστές* (personal computers) είναι οι φθηνότεροι υπολογιστές που συνήθως χρησιμοποιούνται από ένα μόνο χρήστη και έχουν περιορισμένες υπολογιστικές δυνατότητες σε σχέση με άλλους υπολογιστές. Υπάρχουν διάφορες παραλλαγές τους (notebook, laptop, desktop, tower, workstations, network computer). Μια πιο ισχυρή από απόψεως δυνατοτήτων κατηγορία προσωπικών υπολογιστών είναι οι *εξυπηρετητές* (servers). Οι εξυπηρετητές χρησιμοποιούνται για να υποστηρίξουν ένα δίκτυο υπολογιστών που επιτρέπει την κοινή χρήση αρχείων, προγραμμάτων και υλικού όπως οι εκτυπωτές.
  Οι προσωπικοί υπολογιστές βασίζονται στους *μικροϋπολογιστές* (microcomputers), οι οποίοι χρησιμοποιούνται και σε άλλες εφαρμογές, όπως είναι τα *ενσωματωμένα συστήματα* (embedded systems).
- Οι *μίνι-υπολογιστές* (minicomputers) είναι πιο ισχυρά συστήματα που μπορούν να υποστηρίξουν ένα σύνολο χρηστών, οι οποίοι επιτελούν διαφορετικές εργασίες. Συνήθως χρησιμοποιούνται από εταιρείες και οργανισμούς για επεξεργασία πληροφοριών.
- Οι *μεγάλοι υπολογιστές* (mainframes) αποτελούν ισχυρά συστήματα και μπορούν να υποστηρίξουν εκατοντάδες χρήστες «συγχρόνως». Απαιτούν συνήθως δαπανηρό περιβάλλον υποστήριξης, δηλαδή κλιματισμό, ειδικές αίθουσες κλπ.
- Οι *υπερυπολογιστές* (supercomputers) είναι τα πλέον ισχυρά συστήματα και χρησιμοποιούνται κυρίως σε εξειδικευμένες, υπολογιστικά απαιτητικές, εφαρμογές, όπως π.χ. η μετεωρολογία.

Τα όρια των κατηγοριών αυτών δεν είναι σαφή και μετατοπίζονται συνεχώς, εξαιτίας της ραγδαίας τεχνολογικής εξέλιξης στο χώρο των υπολογιστών.

Ανακεφαλαίωση

Κάθε υπολογιστικό σύστημα αποτελείται από το υλικό, δηλαδή τις συσκευές, και το λογισμικό, που το αποτελούν προγράμματα. Λόγω της πολυπλοκότητάς τους, τα υπολογιστικά συστήματα είναι οργανωμένα ιεραρχικά. Στο κατώτερο επίπεδο της ιεραρχίας βρίσκονται τα ηλεκτρονικά κυκλώματα και στο ανώτερο τα προγράμματα του χρήστη. Μεγάλο μέρος του υλικού απαρτίζεται από ολοκληρωμένα κυκλώματα, που έχουν εξελιχθεί ώστε να έχουν πολύ μικρά μεγέθη και μεγάλες ταχύτητες.

Οι «γενιές» των υπολογιστικών συστημάτων είναι πέντε, ξεκινώντας από τους πρώτους πειραματικούς υπολογιστές μέχρι τις μέρες μας. Οι βασικότερες κατηγορίες υπολογιστικών συστημάτων σήμερα είναι οι προσωπικοί υπολογιστές, οι μίνι-υπολογιστές, οι μεγάλοι υπολογιστές και οι υπερυπολογιστές. Κάθε μία από αυτές τις κατηγορίες έχει διαφορετικές υπολογιστικές ικανότητες και απευθύνεται σε διαφορετικές κοινότητες χρηστών.

Γλωσσάριο όρων

Flip-Flop	
Αποθηκευμένο Πρόγραμμα	*Stored Program*
Εξυπηρετητής	*Server*
Κυκλώματα MOS με Συμπληρωματικά Στοιχεία	*Complementary Metal Oxide Semiconductor - CMOS*
Κυκλώματα Μεγάλης Κλίμακας Ολοκλήρωσης	*Large Scale Integration - LSI*
Κυκλώματα Μέσης Κλίμακας Ολοκλήρωσης	*Medium Scale Integration – MSI*
Κυκλώματα Μικρής Κλίμακας Ολοκλήρωσης	*Small Scale Integration - SSI*
Κυκλώματα Πολύ Μεγάλης Κλίμακας Ολοκλήρωσης	*Very Large Scale Integration - VLSI*
Κυκλώματα Συνδεδεμένου Εκπομπού	*Emitter Coupled Logic - ECL*
Κυκλώματα Τρανζίστορ - Τρανζίστορ	*Transistor Transistor Logic - TTL*
Λογικό Κύκλωμα	*Logical Circuit*
Λογισμικό	*Software*
Λογισμικό Εφαρμογών	*Applications Software*
Λογισμικό Συστήματος	*Systems Software*
Μεγάλος Υπολογιστής	*Mainframe*
Μετρητής Προγράμματος	*Program Counter*
Μικρολειτουργία	*Microoperation*
Μικροϋπολογιστής	*Microcomputer*
Μίνι-υπολογιστής	*Minicomputer*
Ολοκληρωμένο Κύκλωμα	*Integrated Circuit - IC*
Προσωπικός Υπολογιστής	*Personal Computer*
Πύλη	*Gate*
Υλικό	*Hardware*
Υπερυπολογιστής	*Supercomputer*
Υπολογιστής Τύπου Von Neumann	
Υπολογιστικό Σύστημα	*Computer System*
Ψηφίδα	*Chip*

Ερωτήσεις

? Ποιες είναι οι συνιστώσες ενός υπολογιστικού συστήματος;

? Τι εννοούμε με τον όρο «Λογισμικό Συστήματος»;

? Ποιες είναι οι διαφορές των ολοκληρωμένων κυκλωμάτων διαφόρων κλιμάκων ολοκλήρωσης;

? Ποια είναι τα χαρακτηριστικά των υπολογιστών της $4^{ης}$ και $5^{ης}$ γενιάς;

Τι μάθαμε σε αυτό το κεφάλαιο

- Κάθε υπολογιστικό σύστημα αποτελείται από το υλικό, δηλαδή τις συσκευές και το λογισμικό, που απαρτίζεται από προγράμματα.
- Τα υπολογιστικά συστήματα είναι οργανωμένα ιεραρχικά. Στο κατώτερο επίπεδο της ιεραρχίας βρίσκονται τα ηλεκτρονικά κυκλώματα και στο ανώτερο τα προγράμματα του χρήστη.
- Τα υπολογιστικά συστήματα έχουν περάσει από διάφορα στάδια εξέλιξης, τα οποία ονομάζονται «γενιές».
- Οι βασικότερες κατηγορίες υπολογιστικών συστημάτων σήμερα είναι οι μικροϋπολογιστές με κυριότερους τους προσωπικούς υπολογιστές, οι μίνι-υπολογιστές, οι μεγάλοι υπολογιστές και οι υπερυπολογιστές.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Π. Τσανάκα, Γ. Φραγκάκη, Αρχιτεκτονική Υπολογιστών, Συμμετρία, 1987.
- Hayes J., Computer Architecture and Organization, McGraw-Hill, 1988.
- Mano M., Computer system Architecture, Prentice-Hall, 1982.
- Patterson D., Hennessy J. Computer Organization & Design, The Hardware/Software Interface, Morgan Kaufmann, 1994.

Κεφάλαιο

Παράσταση και Επεξεργασία Πληροφοριών

Σκοπός του κεφαλαίου αυτού είναι να εξηγήσει πώς παριστάνονται οι πληροφορίες από τον υπολογιστή με τη μορφή 0 και 1.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να χρησιμοποιείς διάφορα συστήματα αρίθμησης και να μετατρέπεις αριθμούς από το ένα στο άλλο.
- Να κάνεις πράξεις στο δυαδικό σύστημα με ακέραιους και κλασματικούς αριθμούς, θετικούς και αρνητικούς.
- Να περιγράφεις τις διάφορες τεχνικές συμπίεσης των δεδομένων.
- Να εξηγείς πώς παριστάνονται ο ήχος, η εικόνα και το video με 0 και 1.

Μαθήματα

2.1 Συστήματα Αρίθμησης

2.2 Παράσταση Αριθμών και Αριθμητικές Πράξεις

2.3 Μέθοδοι Συμπίεσης Δεδομένων

2.4 Ψηφιακή Παράσταση Ήχου, Εικόνας, Video

Μάθημα 2.1 Συστήματα Αρίθμησης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τον τρόπο παράστασης των αριθμών σε διάφορα συστήματα αρίθμησης πέρα από το γνωστό μας δεκαδικό.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό θα μπορείς:

- *Να εξηγείς πώς παριστάνονται οι αριθμοί στα διάφορα συστήματα αρίθμησης.*
- *Να μετατρέπεις αριθμούς από και προς διάφορα συστήματα αρίθμησης.*

Οι υπολογιστές αναπαριστούν όλα τα είδη πληροφορίας ως δυαδικά δεδομένα. Έτσι, για την ευκολότερη και ταχύτερη επεξεργασία των διαφόρων πληροφοριών, οι υπολογιστές χρησιμοποιούν αριθμητικά συστήματα διαφορετικά από το γνωστό μας *δεκαδικό* (decimal) σύστημα και κυρίως το *δυαδικό* (binary).

Αριθμητικά Συστήματα

Κάθε αριθμός N μπορεί να γραφεί με την ακόλουθη μορφή:

$$N = \sum_{i=-n}^{m-1} \alpha_i \cdot \beta^i = \underbrace{\alpha_{m-1} \cdot \beta^{m-1} + \alpha_{m-2} \cdot \beta^{m-2} + \ldots + \alpha_1 \cdot \beta^1 + \alpha_0 \cdot \beta^0}_{\text{ακέραιο μέρος του αριθμού}} + \underbrace{\alpha_{-1} \cdot \beta^{-1} + \alpha_{-2} \cdot \beta^{-2} + \ldots + \alpha_{-n} \cdot \beta^{-n}}_{\text{κλασματικό μέρος του αριθμού}}$$

Με α_i συμβολίζουμε τα ψηφία του αριθμού και με β παριστάνουμε τη βάση του αριθμητικού συστήματος στο οποίο εκφράζεται ο αριθμός. Το ψηφίο α_i πολλαπλασιάζεται με τον αριθμό β^i , γι'αυτό λέμε ότι η *τάξη* (order) του ψηφίου α_i είναι i. Το κλασματικό τμήμα του αριθμού είναι αυτό μετά την υποδιαστολή και είναι μικρότερο του 1. Αν ο αριθμός N έχει m ακέραια ψηφία, οι εκθέτες i παίρνουν τιμές από 0 έως m-1 για το ακέραιο μέρος του και αν τα κλασματικά του ψηφία είναι n, οι εκθέτες i παίρνουν αρνητικές τιμές από -1 έως -n για το κλασματικό του τμήμα.

> Ο δεκαδικός αριθμός 19,278 με τον τρόπο αυτό γράφεται ως $\mathbf{1}\cdot 10^1 + \mathbf{9}\cdot 10^0 + \mathbf{2}\cdot 10^{-1} + \mathbf{7}\cdot 10^{-2} + \mathbf{8}\cdot 10^{-3}$, δηλαδή $\alpha_1=1$, $\alpha_0=9$, $\alpha_{-1}=2$, $\alpha_{-2}=7$, $\alpha_{-3}=8$.

Ένα αριθμητικό σύστημα με βάση β χρειάζεται β διαφορετικά «ψηφία» για την παράσταση των αριθμών, που παίρνουν τις τιμές από 0 έως β-1. Ένας φυσικός αριθμός που έχει m ψηφία, στο σύστημα αυτό μπορεί να πάρει τιμές από 0 έως β^m-1, δηλαδή β^m διαφορετικές τιμές.

Δεκαδικά ψηφία	Δυαδικά ψηφία	Οκταδικά ψηφία	Δεκαεξαδικά ψηφία
0 5 1 6 2 7 3 8 4 9	0 1	0 4 1 5 2 6 3 7	0 8 1 9 2 A 3 B 4 C 5 D 6 E 7 F

Τα συνηθέστερα αριθμητικά συστήματα είναι αυτά που έχουν βάση τους αριθμούς 2 (δυαδικό σύστημα) ,8 (οκταδικό σύστημα, octal system),10 (δεκαδικό σύστημα) και16 (δεκαεξαδικό σύστημα, hexadecimal system). Στον πίνακα βλέπουμε τα ψηφία αυτών των αριθμητικών συστημάτων.

Το δεκαεξαδικό σύστημα χρειάζεται 16 ψηφία για την παράσταση των αριθμών, αλλά το γνωστό μας αριθμητικό σύστημα παρέχει μόνο 10 ψηφία (0-9). Για τα επιπλέον 6 ψηφία χρησιμοποιούμε τους χαρακτήρες A-F, δηλαδή A=10, B=11, C=12, D=13, E=14 και F=15.

> Στο δεκαεξαδικό σύστημα, η ακολουθία ψηφίων «6F3» παριστάνει τον αριθμό $6\cdot16^2 + 15\cdot16^1 + 3\cdot16^0$

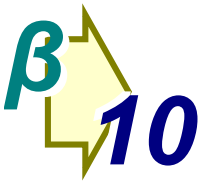
Στην καθημερινή μας ζωή χρησιμοποιούμε το δεκαδικό σύστημα και είμαστε εξοικειωμένοι με αυτό, έτσι παριστάνουμε τους αριθμούς μόνο με τα ψηφία τους, π.χ. λέμε 15 και όχι $1\cdot10^1+5\cdot10^0$. Το ίδιο μπορούμε να κάνουμε και με οποιοδήποτε άλλο αριθμητικό σύστημα, αρκεί να δηλώνουμε το σύστημα αυτό. Ο προσδιορισμός του συστήματος γίνεται συνήθως με ένα δείκτη που συνοδεύει τον αριθμό και δηλώνει τη βάση του αριθμητικού συστήματος.

Το δεκαδικό σύστημα δεν είναι το μόνο που έχει χρησιμοποιηθεί από τον άνθρωπο. Οι Βαβυλώνιοι χρησιμοποιούσαν αριθμητικό σύστημα με βαση το 60.

> Η ακολουθία ψηφίων «321» παριστάνει διαφορετικούς αριθμούς ανάλογα με το σύστημα αρίθμησης που θα δηλώσουμε. Στο δεκαδικό σύστημα θα γράψουμε $321_{(10)}$ και θα εννοούμε $3\cdot10^2+2\cdot10^1+1\cdot10^0$, ενώ στο οκταδικό σύστημα θα γράψουμε $321_{(8)}$ και θα εννοούμε $3\cdot8^2+2\cdot8^1+1\cdot8^0$. Στο δεκαεξαδικό σύστημα θα γράψουμε $321_{(16)}$ και θα εννοούμε $3\cdot16^2+2\cdot16^1+1\cdot16^0$.
>
> Ο αριθμός $1101_{(2)}$ είναι γραμμένος στο δυαδικό σύστημα, έτσι αντιστοιχεί στην έκφραση $1\cdot2^3 + 1\cdot2^2 + 0\cdot2^1 + 1\cdot2^0$.

Τα ψηφία ενός αριθμού γραμμένου στο δυαδικό σύστημα ονομάζονται bits (**b**inary dig**its**, δυαδικά ψηφία). Το πιο αριστερό ψηφίο του αριθμού ονομάζεται *περισσότερο σημαντικό ψηφίο* (Most Significant Bit, MSB), γιατί πολλαπλασιάζεται με το μεγαλύτερο συντελεστή, και το πιο δεξιό ψηφίο του αριθμού ονομάζεται *λιγότερο σημαντικό ψηφίο* (Least Significant Bit, LSB), γιατί πολλαπλασιάζεται με το μικρότερο συντελεστή.

Μετατροπή αριθμών από ένα αριθμητικό σύστημα σε άλλο

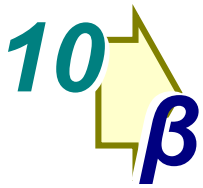


Η μετατροπή ενός αριθμού από ένα αριθμητικό σύστημα με βάση β προς το δεκαδικό σύστημα είναι πολύ απλή: υπολογίζουμε την τιμή της παράστασης $\alpha_{m-1}\cdot\beta^{m-1} + \ldots + \alpha_1\cdot\beta^1 + \alpha_0\cdot\beta^0 + \alpha_{-1}\cdot\beta^{-1} + \ldots + \alpha_{-n}\cdot\beta^{-n}$.

> Ο δυαδικός αριθμός $10011_{(2)}$ στο δεκαδικό σύστημα έχει την τιμή $1\cdot2^4 + 0\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 1\cdot2^0 = 16+2+1 = 19_{(10)}$.
>
> Ο οκταδικός αριθμός $7123{,}35_{(8)}$ στο δεκαδικό σύστημα έχει την τιμή $7\cdot8^3 + 1\cdot8^2 + 2\cdot8^1 + 3\cdot8^0 + 3\cdot8^{-1} + 5\cdot8^{-2} = 3584 + 64 +16 + 3 + 0{,}375 + 0{,}0781 = 3667{,}4531_{(10)}$.
>
> Ο δεκαεξαδικός αριθμός $FC27_{(16)}$ αντιστοιχεί στο δεκαδικό $15\cdot16^3 + 12\cdot16^2 + 2\cdot16^1 + 7\cdot16^0 = 61440 + 3072 + 32 + 7 = 64551_{(10)}$.

Πιο πολύπλοκη είναι η διαδικασία μετατροπής ενός αριθμού από το δεκαδικό σύστημα σε ένα άλλο σύστημα αρίθμησης με βάση β. Η μετατροπή γίνεται χωριστά για το ακέραιο και χωριστά για το κλασματικό μέρος.

Για να μετατρέψουμε το ακέραιο μέρος του αριθμού Α σε βάση β, κάνουμε διαδοχικές διαιρέσεις του ακεραίου μέρους του Α με τον αριθμό β. Η διαδικασία μετατροπής είναι η εξής:



(1) Διαιρούμε το δεκαδικό αριθμό με το β, και παίρνουμε το πηλίκο Π και το υπόλοιπο Υ. Το υπόλοιπο είναι μία τιμή από 0 έως β-1 και αποτελεί το δεξιότερο ψηφίο του αριθμού.

(2) Διαιρούμε το πηλίκο Π πάλι με το β, και παίρνουμε ένα νέο πηλίκο Π και υπόλοιπο Υ. Γράφουμε το υπόλοιπο (που πάλι είναι μία τιμή από 0 έως β-1) στα αριστερά του αριθμού.

(3) Επαναλαμβάνουμε το βήμα (2) μέχρι το πηλίκο Π να γίνει 0.

Ας δούμε πώς μετατρέπεται ο αριθμός $A=53_{(10)}$ στο δυαδικό σύστημα (β=2):

		Π	**Υ**	**Χ**
53	*Διαιρούμε το 53 με το 2* →	26	1	**1**
26	*Διαιρούμε το 26 με το 2* →	13	0	**0**1
13	*Διαιρούμε το 13 με το 2* →	6	1	**1**01
6	*Διαιρούμε το 6 με το 2* →	3	0	**0**101
3	*Διαιρούμε το 3 με το 2* →	1	1	**1**0101
1	*Διαιρούμε το 1 με το 2* →	0	1	**1**10101
0	→ **$53_{(10)} = 110101_{(2)}$**			

Ας δούμε και τη μετατροπή του αριθμού 312 στο οκταδικό σύστημα:

		Π	**Υ**	**Χ**
312	*Διαιρούμε το 312 με το 8* →	39	0	**0**
39	*Διαιρούμε το 39 με το 8* →	4	7	**7**0
4	*Διαιρούμε το 4 με το 8* →	0	4	**47**0
0	→ **$312_{(10)} = 470_{(8)}$**			

Για να μετατρέψουμε το κλασματικό μέρος ενός αριθμού Α από το δεκαδικό σύστημα σε ένα άλλο σύστημα αρίθμησης με βάση β, κάνουμε διαδοχικούς πολλαπλασιασμούς του κλασματικού μέρους του Α με τη βάση β. Το κλασματικό μέρος στο νέο σύστημα αρίθμησης μπορεί να έχει άπειρα ψηφία, γι' αυτό καθορίζουμε από πριν το μέγιστο αριθμό ψηφίων Ν που θα υπολογίσουμε για το νέο σύστημα αρίθμησης. Η διαδικασία μετατροπής είναι η ακόλουθη:

(1) Χρησιμοποιούμε μόνο το κλασματικό μέρος του Α.

(2) Πολλαπλασιάζουμε τον Α με τη βάση β. Το αποτέλεσμα έχει ακέραιο μέρος Μ και κλασματικό μέρος Κ. Το Μ (που έχει τιμή από 0 έως β-1) είναι το αριστερότερο ψηφίο του νέου κλασματικού μέρους Υ.

(3) Πολλαπλασιάζουμε το κλασματικό μέρος Κ με το β και παίρνουμε ένα νέο αποτέλεσμα με ακέραιο μέρος Μ (με τιμή από 0 έως β-1) και κλασματικό μέρος Κ.

(4) Γράφουμε το Μ στα δεξιά του νέου κλασματικού μέρους Υ.

(5) Επαναλαμβάνουμε τα βήματα (3), (4) έως ότου το Μ να γίνει 0 ή να έχουμε υπολογίσει Ν ψηφία.

Ας υπολογίσουμε την τιμή του κλασματικού αριθμού $0{,}625_{(10)}$ στο δυαδικό σύστημα:

		M	K	Y
0,625	$0{,}625 \times 2 = 1{,}25$	1	0,25	0,**1**
0,25	$0{,}25 \times 2 = 0{,}5$	0	0,5	0,1**0**
0,5	$0{,}5 \times 2 = 1$	1	0	0,10**1**
0	$\rightarrow$ $\mathbf{0{,}625_{(10)} = 0{,}101_{(2)}}$			

Επίσης ας υπολογίσουμε την τιμή του κλασματικού αριθμού $0{,}171875_{(10)}$ στο δεκαξαδικό σύστημα:

		M	K	Y
0, 171875	$0{,}171875 \times 16 = 2{,}75$	2	0,75	0,**2**
0,75	$0{,}75 \times 16 = 12$	12	0	0,2**C**
0	$\rightarrow$ $\mathbf{0{,}171875_{(10)} = 0{,}2C_{(16)}}$			

Ας υπολογίσουμε και την τιμή του κλασματικού αριθμού $0{,}4_{(10)}$ στο οκταδικό σύστημα. Καθορίζουμε από πριν ότι θα υπολογίσουμε το πολύ 5 οκταδικά ψηφία.

		M	K	Y
0,4	$0{,}4 \times 8 = 3{,}2$	3	0,2	0,**3**
0,2	$0{,}2 \times 8 = 1{,}6$	1	0,6	0,3**1**
0,6	$0{,}6 \times 8 = 4{,}8$	4	0,8	0,31**4**
0,8	$0{,}8 \times 8 = 6{,}4$	6	0,4	0,314**6**
0,4	$0{,}4 \times 8 = 3{,}2$	3	0,2	0,3146**3**
0,2	$\rightarrow$ $\mathbf{0{,}4_{(10)} \cong 0{,}31463_{(8)}}$			

Εδώ σταματήσαμε τον υπολογισμό παρότι το Μ δεν είναι 0, γιατί φθάσαμε στο μέγιστο αριθμό κλασματικών ψηφίων που επιτρέπουμε. Αν προσέξουμε τις τιμές που παίρνει το Μ, παρατηρούμε ότι μετά από τον υπολογισμό του 4^{ου} κλασματικού ψηφίου το Μ παίρνει την αρχική του τιμή που ήταν 0,4. Έτσι μπορούμε να συμπεράνουμε ότι στο οκταδικό σύστημα ο Α είναι ένας περιοδικός αριθμός, ο $0{,}31463146\ldots_{(8)}$ ή αλλιώς $0{,}\overline{3146}_{(8)}$.

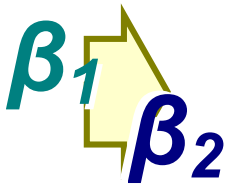
Όταν σταματάμε τον υπολογισμό του αριθμού μετά από Ν κλασματικά ψηφία, αν και το Μ δεν είναι 0, κάνουμε *αποκοπή* (truncation) του αριθμού. Για να είναι ο υπολογισμός μας όσο πιο ακριβής γίνεται, μπορούμε να κάνουμε *στρογγυλοποίηση* (rounding). Στη στρογγυλοποίηση υπολογίζουμε άλλο ένα κλασματικό ψηφίο. Αν αυτό είναι μικρότερο από το ½ της βάσης, τότε αφήνουμε τον αριθμό όπως είναι, με Ν

ψηφία. Αν όμως το επιπλέον ψηφίο είναι μεγαλύτερο ή ίσο από το ½ της βάσης, τότε αυξάνουμε το τελευταίο (το N-οστό) ψηφίο κατά 1. Έτσι το σφάλμα του υπολογισμού είναι πιο μικρό.

> Κατά τη μετατροπή του αριθμού $0,4_{(10)}$ στο δεκαεξαδικό σύστημα, ολοκληρώσαμε τον υπολογισμό μετά από 5 ψηφία. Το επόμενο ψηφίο που θα υπολογίζαμε είναι το 1, που είναι μικρότερο από το 4 (το ½ της βάσης), έτσι και μετά από τη στρογγυλοποίηση ο αριθμός μένει ο ίδιος.
>
> Αν όμως κρατούσαμε μόνο N=3 κλασματικά ψηφία στρογγυλοποιώντας το αποτέλεσμα, θα υπολογίζαμε και το $4^{ο}$ ψηφίο που έχει την τιμή 6 > 4. Θα αυξάναμε λοιπόν το $3^{ο}$ ψηφίο κατά 1, και ο αριθμός θα ήταν τελικά ο $0,315_{(8)}$.

Για να μετατρέψουμε από το δεκαδικό σύστημα αρίθμησης σε άλλο έναν αριθμό που έχει και ακέραιο και κλασματικό μέρος, μετατρέπουμε ξεχωριστά τα δύο μέρη του με τον τρόπο που είδαμε και μετά συνδυάζουμε τα αποτελέσματα.

Η μετατροπή ενός αριθμού από ένα σύστημα με βάση β_1 σε ένα άλλο σύστημα με βάση β_2 γίνεται εύκολα άν χρησιμοποιήσουμε ενδιάμεσα το δεκαδικό σύστημα: μετατρέπουμε πρώτα τον αριθμό με βάση β_1 στο δεκαδικό σύστημα, και στη συνέχεια τον μετατρέπουμε από το δεκαδικό σύστημα στο σύστημα με βάση β_2. Η μέθοδος αυτή είναι πιο εύκολη από την απευθείας μετατροπή, γιατί είμαστε πιο εξοικειωμένοι με υπολογισμούς στο δεκαδικό σύστημα.

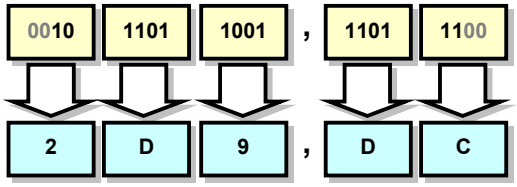


Μια ειδική περίπτωση όμως είναι η μετατροπή μεταξύ του δυαδικού και του οκταδικού ή του δεκαεξαδικού συστήματος. Οι μετατροπές αυτές είναι ιδιαίτερα εύκολες, γιατί οι βάσεις των δύο συστημάτων, το 8 και το 16, είναι δυνάμεις του 2.

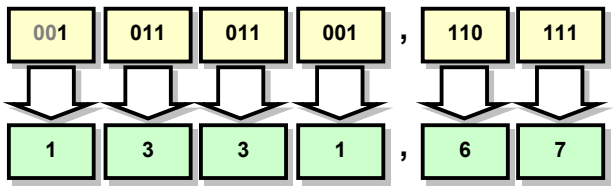
Για να μετατρέψουμε ένα δυαδικό αριθμό στο δεκαεξαδικό σύστημα, χωρίζουμε τα ψηφία του σε τετράδες ξεκινώντας από την υποδιαστολή που χωρίζει ακέραιο και κλασματικό μέρος, και προχωρώντας προς τα «άκρα» του αριθμού. Κάθε τέτοια τετράδα αντιστοιχεί σε ένα μονοψήφιο δεκαεξαδικό αριθμό, και την αντικαθιστούμε με το ψηφίο αυτό. Η μετατροπή ενός δεκαεξαδικού αριθμού σε δυαδικό γίνεται αντικαθιστώντας κάθε ψηφίο του αριθμού με τον αντίστοιχο τετραψήφιο δυαδικό αριθμό.

Η μετατροπή από το δυαδικό σύστημα προς το οκταδικό και αντίστροφα γίνεται με τον ίδιο τρόπο, αλλά χωρίζουμε τα δυαδικά ψηφία σε τριάδες αντί για τετράδες.

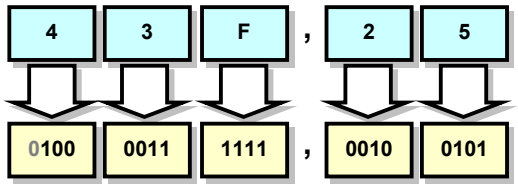
> Για να μετατραπεί σε δεκαεξαδικό, χωρίζουμε το δυαδικό αριθμό $1011011001,11011_{(2)}$ πρώτα σε τετράδες ξεκινώντας από την υποδιαστολή. Στα «άκρα» του αριθμού προσθέτουμε όσα μηδενικά είναι απαραίτητα, έτσι ώστε να συμπληρωθούν οι τετράδες. Στη συνέχεια αντικαθιστούμε κάθε τετράδα με το αντίστοιχο δεκαεξαδικό ψηφίο. Π.χ. η αριστερότερη τετράδα που είναι 0010 ισοδυναμεί με το ψηφίο 2, ενώ η επόμενη τετράδα 1101 που έχει τιμή 13 ισοδυναμεί με το ψηφίο D. Ο ισοδύναμος δεκαεξαδικός αριθμός είναι ο $2D9,DC_{(16)}$.



> Για να μετατρέψουμε τον ίδιο αριθμό σε οκταδικό τον χωρίζουμε σε τριάδες, προσθέτοντας και πάλι στα άκρα του μηδενικά, αν χρειαστεί. Κάθε τριάδα αντικαθίσταται από το αντίστοιχο οκταδικό ψηφίο. Π.χ. η δεξιότερη τριάδα που είναι 111 έχει την τιμή 7 ($1 \cdot 2^2 + 1 \cdot 2 + 1 = 4+2+1 = 7$) και αντικαθίσταται με το

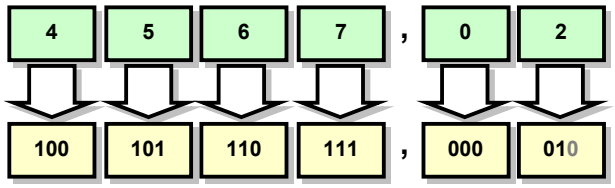


ψηφίο αυτό. Ο αντίστοιχος οκταδικός αριθμός είναι ο $1331,67_{(8)}$.



Αντίστροφα, η μετατροπή του δεκαεξαδικού αριθμού $43F,25_{(16)}$ γίνεται αντικαθιστώντας τα ψηφία του με τον αντίστοιχο τετραψήφιο δυαδικό αριθμό. Π.χ. το ψηφίο 4 θα αντικατασταθεί από τον δυαδικό αριθμό 0100 που έχει την τιμή 4. Ο δυαδικός αριθμός που προκύπτει είναι ο $10000111111,00100101_{(2)}$.

Στον οκταδικό αριθμό $4567,02_{(8)}$ θα αντικαταστήσουμε κάθε ψηφίο με τον αντίστοιχο *τριψήφιο* δυαδικό αριθμό, για να πάρουμε το δυαδικό αριθμό $100101110111,000010_{(2)}$.



Τα μηδενικά στα «άκρα» των δυαδικών αριθμών (όπως και σε όλα τα συστήματα αρίθμησης) μπορούν να παραλειφθούν.

Ανακεφαλαίωση

Ένα αριθμητικό σύστημα με βάση β παριστάνει κάθε αριθμό ως άθροισμα δυνάμεων του αριθμού β. Τα πιο γνωστά συστήματα είναι το δυαδικό, το οκταδικό, το δεκαδικό και το δεκαεξαδικό. Η μετατροπή ενός ακέραιου ή πραγματικού αριθμού από το δεκαδικό σύστημα σε ένα άλλο σύστημα με βάση β γίνεται με διαδοχικές διαιρέσεις ή πολλαπλασιασμούς με τη βάση β, αντίστοιχα. Οι μετατροπές μεταξύ άλλων συστημάτων αρίθμησης γίνονται για ευκολία μέσω του δεκαδικού συστήματος. Η μόνη εξαίρεση είναι οι μετατροπές μεταξύ δυαδικού και οκταδικού ή δεκαεξαδικού συστήματος, που γίνονται πολύ εύκολα απευθείας.

Γλωσσάριο όρων

Αποκοπή	*Truncation*
Δεκαδικό Σύστημα	*Decimal System*
Δεκαεξαδικό Σύστημα	*Hexadecimal System*
Δυαδικό Σύστημα	*Binary System*
Δυαδικό Ψηφίο	*Binary Digit - Bit*
Λιγότερο Σημαντικό Ψηφίο	*Least Significant Bit - LSB*
Οκταδικό Σύστημα	*Octal System*
Περισσότερο Σημαντικό Ψηφίο	*Most Significant Bit - MSB*
Στρογγυλοποίηση	*Rounding*
Τάξη	*Order*

Ερωτήσεις

? Πόσα ψηφία χρειάζεται ένα σύστημα αρίθμησης με βάση β;

? Πώς μετατρέπουμε έναν ακέραιο δυαδικό αριθμό στο δεκαδικό σύστημα;

? Πώς μετατρέπουμε ένα δεκαδικό αριθμό στο δυαδικό σύστημα;

? Τι είναι η αποκοπή και η στρογγυλοποίηση;

? Πώς γίνεται η μετατροπή μεταξύ δυαδικού και οκταδικού συστήματος;

Μάθημα 2.2 Παράσταση Αριθμών και Χαρακτήρων

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει πώς παριστάνονται οι αριθμοί σε ένα υπολογιστή και τη διαδικασία με την οποία γίνονται οι αριθμητικές πράξεις μεταξύ τους.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό θα μπορείς:

- *Να εξηγείς πώς γίνονται οι αριθμητικές πράξεις μεταξύ ακεραίων αριθμών*
- *Να υπολογίζεις την παράσταση προσημασμένων ακεραίων αριθμών και να εκτελείς πράξεις μεταξύ τους*
- *Να υπολογίζεις την παράσταση πραγματικών αριθμών και να εκτελείς πράξεις μεταξύ τους*
- *Να βρίσκεις την παράσταση χαρακτήρων με την κωδικοποίηση ASCII*

Αριθμητικές πράξεις

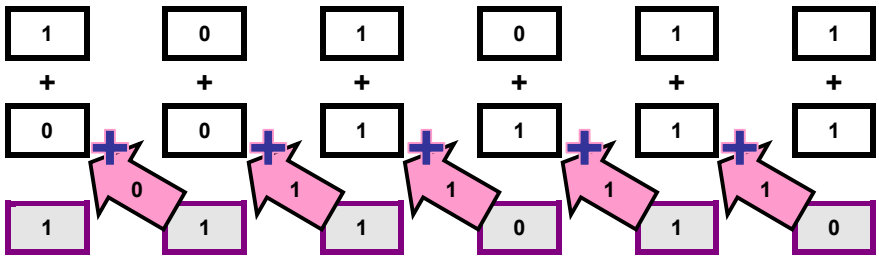
Στο δυαδικό σύστημα οι αριθμητικές πράξεις γίνονται όπως και στο δεκαδικό σύστημα. Προσθέσεις και αφαιρέσεις γίνονται από δεξιά προς τα αριστερά, και χρησιμοποιούμε κρατούμενα ή δανεικά ψηφία αντίστοιχα.

Για την πρόσθεση αριθμών με n bits, ξεκινάμε από τα δεξιά, προσθέτοντας τα δύο λιγότερα σημαντικά bits των αριθμών και συνεχίζουμε προς τα αριστερά. Αν οι αριθμοί είναι οι $x_n x_{n-1} \ldots x_i \ldots x_1$ και $y_n y_{n-1} \ldots y_i \ldots y_1$, ξεκινάμε από την πρόσθεση x_1+y_1, η οποία μας δίνει το ψηφίο z_1 του αποτελέσματος και το κρατούμενο K_1. Στη συνέχεια προσθέτουμε τα ψηφία x_2, y_2 και το κρατούμενο K_1 για να πάρουμε το ψηφίο z_2 του αποτελέσματος και το κρατούμενο K_2, κλπ. Γενικά το κρατούμενο K_{i-1} που πιθανώς θα προκύψει από κάποια πρόσθεση προωθείται και προστίθεται με το αμέσως αριστερότερο ζεύγος ψηφίων x_i και y_i των αριθμών. Αντίστοιχα στην αφαίρεση λαμβάνουμε τα αντίστοιχα ζεύγη των ψηφίων x_i και y_i, κάνουμε την πράξη και προωθούμε το δανεικό ψηφίο Δ_i στο επόμενο ζεύγος ψηφίων x_{i+1} και y_{i+1}.

Οι υπολογισμοί που κάνουμε για αριθμούς των n bits αφορούν λοιπόν συνήθως 3 bits, γιατί έχει προστεθεί και το κρατούμενο ή το δανεικό ψηφίο. Στον πίνακα βλέπουμε τα αποτελέσματα της πρόσθεσης και αφαίρεσης δύο ψηφίων με κρατούμενο ή δανεικό ψηφίο αντίστοιχα.

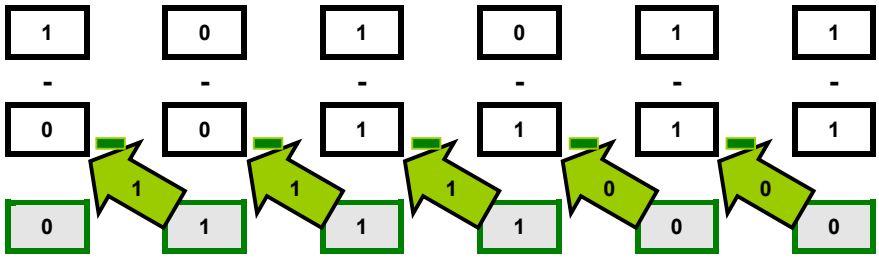
x_i	y_i	K_{i-1} ή Δ_{i-1}	$x_i+y_i+K_{i-1}$	K_i	$x_i-y_i-\Delta_{i-1}$	Δ_i
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	0	1	1
0	1	1	0	1	0	1
1	0	0	1	0	1	0
1	0	1	0	1	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Θέλουμε να προσθέσουμε τους αριθμούς 43 και 15, αλλά με τη δυαδική τους αναπαράσταση. Ο αριθμός 43 στο δυαδικό σύστημα είναι ο $101011_{(2)}$ ενώ ο 15 παριστάνεται ως $001111_{(2)}$.



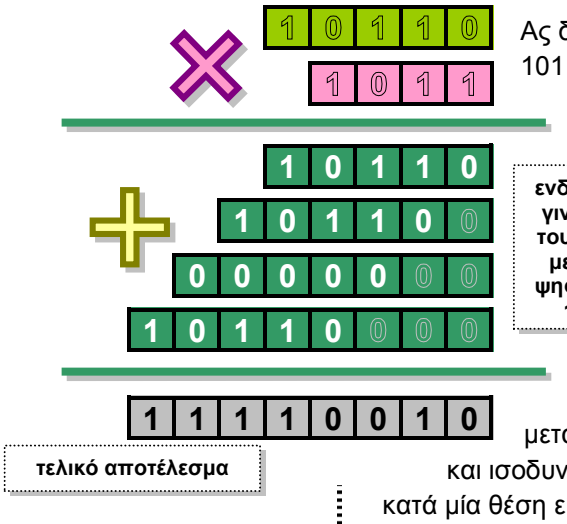
Βλέπουμε στο σχήμα πώς το κρατούμενο από κάθε ζεύγος ψηφίων προωθείται στο επόμενο. Το αποτέλεσμα είναι ο δυαδικός αριθμός $111010_{(2)}$ δηλαδή ο αριθμός 58.

Αν θέλουμε να κάνουμε την αφαίρεση 43-15 αντίστοιχα θα έχουμε:



Το αποτέλεσμα είναι ο δυαδικός αριθμός $011100_{(2)}$ δηλαδή ο δεκαδικός 28. Και εδώ βλέπουμε τα δανεικά ψηφία του κάθε ζεύγους που μεταφέρονται στο επόμενο επίπεδο.

Ο πολλαπλασιασμός και η διαίρεση δυαδικών αριθμών γίνονται με διαδοχικές προσθέσεις και αφαιρέσεις αντίστοιχα.



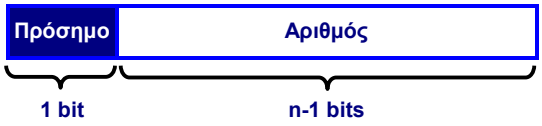
Ας δούμε πώς πολλαπλασιάζουμε τους αριθμούς $22_{(10)}$ = $10110_{(2)}$ και $11_{(10)}$ = $1011_{(2)}$.

Για να δημιουργήσουμε κάθε ένα από τα μερικά γινόμενα, πολλαπλασιάζουμε τον «πάνω» αριθμό (τον 10110) με το αντίστοιχο ψηφίο του άλλου αριθμού (του 1011) εργαζόμενοι από τα δεξιά προς τα αριστερά. Κάθε μερικό γινόμενο είναι μετακινημένο κατά μία θέση αριστερά σε σχέση με το προηγούμενο, ώστε η πρόσθεση όλων μαζί να μας δώσει το σωστό αποτέλεσμα, και έχει συμπληρωθεί από τα δεξιά με μηδενικά. Αυτή η μετακίνηση ονομάζεται *ολίσθηση προς τα αριστερά* (left shift) και ισοδυναμεί με πολλαπλασιασμό: ολίσθηση προς τα αριστερά κατά μία θέση είναι ισοδύναμη με πολλαπλασιασμό του αριθμού επί 2.

Παράσταση αριθμών

Η μνήμη κάθε υπολογιστή είναι οργανωμένη σε *λέξεις* (words), δηλαδή ομάδες των n bits (το n είναι συνήθως ένα πολλαπλάσιο του 8). Το n ονομάζεται *μήκος λέξης* (word length) του υπολογιστή. Κάθε αριθμός θα καταλαμβάνει χώρο όσο μία λέξη της μνήμης του υπολογιστή.

Με N δυαδικά ψηφία μπορούμε να παραστήσουμε 2^N το πλήθος διαφορετικούς αριθμούς, τους 0 … 2^N-1. Στην περίπτωση που θέλουμε με n δυαδικά ψηφία να παραστήσουμε προσημασμένους ακέραιους αριθμούς, τότε εκμεταλλευόμαστε το αριστερότερο bit (δηλαδή το MSB) του αριθμού, στο οποίο κωδικοποιούμε το πρόσημό του. Αν το πρόσημο έχει την τιμή 0, τότε ο αριθμός είναι θετικός, ενώ αν έχει την τιμή 1 είναι αρνητικός. Με τα υπόλοιπα N-1 δυαδικά ψηφία κωδικοποιούμε την απόλυτη τιμή του αριθμού, δηλαδή το *μέτρο* του.



> Οι αριθμοί $01110010_{(2)}$ και $00000001_{(2)}$ είναι θετικοί, ενώ οι αριθμοί $11110010_{(2)}$ και $11100001_{(2)}$ είναι αρνητικοί.

Θετικοί αριθμοί

Ένα θετικό αριθμό τον παριστάνουμε θέτοντας το πιο σημαντικό bit (δηλαδή το πρόσημο) στην τιμή 0, και τα υπόλοιπα n-1 bits στην τιμή του μέτρου του, δηλαδή στην τιμή του αριθμού. Επειδή ο μεγαλύτερος ακέραιος αριθμός που παριστάνεται με n-1 bits είναι ο 2^{n-1}-1, η τιμή του αριθμού δεν μπορεί να ξεπερνά το όριο αυτό.

> Σε έναν υπολογιστή όπου το μήκος λέξης είναι 8, ο αριθμός 23 θα παρασταθεί ως 00010111. Το αριστερότερο bit δηλώνει ότι ο αριθμός είναι θετικός, και τα υπόλοιπα bits περιέχουν τον αριθμό 23. Ο μεγαλύτερος θετικός αριθμός που μπορεί να αποθηκευθεί με 8 bits είναι ο 01111111, δηλαδή ο 127 (2^{8-1}-1 = 2^7-1 = 128-1).

Αρνητικοί αριθμοί

Υπάρχουν τρεις διαφορετικοί τρόποι για να κωδικοποιήσουμε τους αρνητικούς προσημασμένους αριθμούς στα υπόλοιπα n-1 bits:

- Η παράσταση **μέτρου**
- Η παράσταση **συμπληρώματος ως προς 1**
- Η παράσταση **συμπληρώματος ως προς 2**

Στη συνέχεια θα περιοριστούμε στην παράσταση συμπληρώματος ως προς 2.

> Η παράσταση συμπληρώματος ως προς 2 είναι αυτή που χρησιμοποιείται περισσότερο, γιατί διευκολύνει και απλοποιεί πολύ την εκτέλεση των αριθμητικών πράξεων και για τους θετικούς και για τους αρνητικούς αριθμούς.

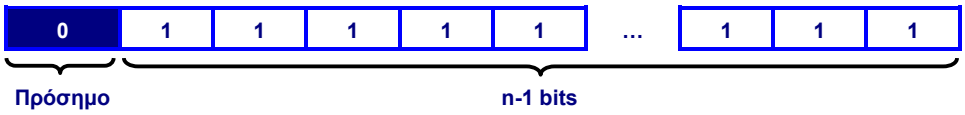
Εάν το πιο σημαντικό ψηφίο του αριθμού είναι 1, τότε ο αριθμός είναι αρνητικός. Για να βρούμε το μέτρο του αριθμού πρέπει να υπολογίσουμε το συμπλήρωμα ως προς 2 και των n ψηφίων του (δηλαδή λαμβάνουμε υπόψη και το πρόσημο). Το συμπλήρωμα ως προς 2 ενός δυαδικού αριθμού βρίσκεται, εάν αντικαταστήσουμε το 0 με 1 και το 1 με 0 και στη συνέχεια προσθέσουμε 1.

Για να μετατρέψουμε ένα αρνητικό στην παράσταση συμπληρώματος του 2, ακολουθούμε παρόμοια διαδικασία: γράφουμε το μέτρο του σε δυαδική μορφή, αντικαθιστούμε το 0 με 1 και το 1 με 0 και στη συνέχεια προσθέτουμε 1. Αν δεν

υπάρχει ήδη ως κρατούμενο, τοποθετούμε στα αριστερά του αριθμού το ψηφίο 1 του προσήμου.

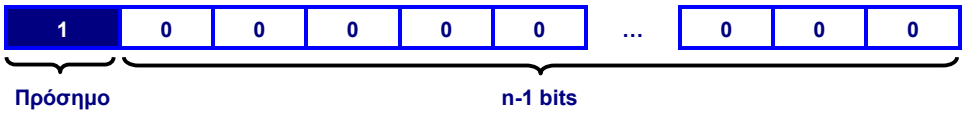
> Για να βρούμε την παράσταση συμπληρώματος ως προς 2 του αριθμού -17 σε ένα υπολογιστή με μήκος λέξης 16 bits, αρχικά θα γράψουμε τον αντίστοιχο θετικό (17) σε δυαδική μορφή, δηλαδή 0000000000010001. Στη συνέχεια θα αντικαταστήσουμε το 0 με 1 και το 1 με 0 στον αριθμό αυτό, και θα πάρουμε 1111111111101110. Στον αριθμό αυτό θα προσθέσουμε τον 1. Η τελική του παράσταση θα είναι λοιπόν 1111111111101111.
>
> Για να βρούμε την τιμή που παριστάνει ο αριθμός 11100110 θα υπολογίσουμε το συμπλήρωμά του ως προς 2 *μαζί με το πρόσημο*. Αρχικά αντιστρέφουμε όλα τα ψηφία του και παίρνουμε 00011001. Μετά προσθέτουμε τον αριθμό 1, και έχουμε 00011001 + 1 = 00011010. Άρα το μέτρο του αριθμού είναι το $00011010_{(2)} = 26_{(10)}$ και ο αριθμός είναι ο -26.

Ο μέγιστος θετικός αριθμός που μπορεί να παρασταθεί στο σύστημα αυτό με λέξη μήκους n bits έχει την παράσταση:



και είναι ο $2^{n-1}-1$

Ο ελάχιστος αρνητικός που μπορεί να παρασταθεί με μήκος λέξης n bits έχει την παράσταση:



Για να βρούμε την τιμή του αριθμού αυτού θα υπολογίσουμε το συμπλήρωμά του ως προς 2. Αντιστρέφουμε τα ψηφία του και παίρνουμε τον αριθμό $01111\ldots1111_{(2)}$ και μετά προσθέτουμε 1, για να πάρουμε τον $10000\ldots000_{(2)} = 2^{n-1}$. Άρα ο ελάχιστος αριθμός είναι ο -2^{n-1}.

Η παράσταση συμπληρώματος του 2 έχει από ό,τι βλέπουμε μία ιδιαιτερότητα: ο μικρότερος αρνητικός αριθμός που μπορούμε να παραστήσουμε έχει μεγαλύτερη απόλυτη τιμή (2^{n-1}) από το μεγαλύτερο θετικό ($2^{n-1}-1$). Αυτό συμβαίνει γιατί ο αντίστοιχός του θετικός, ο 2^{n-1}, χρειάζεται και το ψηφίο του προσήμου για να παρασταθεί.

> Ας μην ξεχνάμε ότι οι έννοιες «συμπλήρωμα ως προς 2» και «παράσταση συμπληρώματος ως προς 2» είναι διαφορετικές. Το συμπλήρωμα ως προς 2 ενός αριθμού είναι το αποτέλεσμα της αντιστροφής των ψηφίων του αριθμού από 0 σε 1 και από 1 σε 0, και της πρόσθεσης σε αυτόν του 1.Η παράσταση συμπληρώματος ως προς δύο χρησιμοποιεί το συμπλήρωμα ως προς 2 για να παραστήσει τους αρνητικούς αριθμούς.

Όπως στο δυαδικό σύστημα υπάρχει η παράσταση συμπληρώματος ως προς 2, έτσι και στο δεκαδικό σύστημα υπάρχει και η παράσταση συμπληρώματος ως προς 10.

Πράξεις προσημασμένων ακεραίων αριθμών

Η παράσταση του συμπληρώματος ως προς 2 έχει ένα πολύ μεγάλο πλεονέκτημα: οι πράξεις μεταξύ των αριθμών γίνονται απευθείας, χωρίς να χρειάζεται μετατροπή τους, ανεξάρτητα από το πρόσημό τους. Έτσι η διαδικασία της πρόσθεσης που είδαμε στην αρχή του μαθήματος εφαρμόζεται αυτούσια και σε προσημασμένους αριθμούς.

00010101 (21)	11001010 (-54)	00010010 (18)	11111101 (-3)
+	+	+	+
00110011 (51)	01000100 (68)	10001101 (-115)	11110010 (-14)
01001000 (72)	100001110 (14)	10011111 (-97)	111101111 (-17)

Στο δεύτερο άθροισμα, το αποτέλεσμα της πρόσθεσης έχει και ένα επιπλέον bit, γιατί $11001010_{(2)} + 01000100_{(2)} = 100001110_{(2)}$. Αυτό το επιπλέον bit το αγνοούμε και παίρνουμε το σωστό αποτέλεσμα. Το ίδιο ισχύει και για το τέταρτο άθροισμα, αφού $11111101_{(2)} + 11110010_{(2)} = 111101111_{(2)}$.

Η μόνη περίπτωση στην εκτέλεση των πράξεων που χρειάζεται προσοχή είναι όταν το αποτέλεσμα μίας πράξης είναι πολύ μεγάλο ή πολύ μικρό και δεν μπορεί να παρασταθεί με το πλήθος των bits που έχουμε στη διάθεσή μας. Τότε λέμε ότι η πράξη προκάλεσε *υπερχείλιση* (overflow) του αποτελέσματος.

10010010	-110
+	
11011111	-33
1 01110001	113

αγνοείται

Το άθροισμα των αριθμών -110 και -33, που είναι -143, δεν μπορεί να παρασταθεί με 8 bits, γιατί ο μικρότερος αριθμός που μπορεί να παρασταθεί με 8 bits είναι ο -128. Αν προσθέσουμε αυτούς τους δύο αριθμούς, όπως βλέπουμε και στο σχήμα, το αποτέλεσμα είναι λανθασμένο.

Οι υπολογιστές διαθέτουν εσωτερικούς μηχανισμούς για να εντοπίζουν τις περιπτώσεις που μία αριθμητική πράξη προκαλεί υπερχείλιση του αποτελέσματός της.

Παράσταση πραγματικών αριθμών

Εκτός από τους ακέραιους αριθμούς, θέλουμε να παραστήσουμε στον υπολογιστή και πραγματικούς αριθμούς, δηλαδή αριθμούς με ακέραιο και κλασματικό μέρος.

Παράσταση σταθερής υποδιαστολής

Ένας απλός τρόπος για την παράσταση των πραγματικών αριθμών με λέξεις μήκους n bits είναι να μοιράσουμε τα ψηφία του αριθμού μεταξύ του ακέραιου και του κλασματικού μέρους, δηλαδή να παριστάνουμε το ακέραιο μέρος του αριθμού με n_1 bits και το κλασματικό μέρος του με n_2 ψηφία, με $n_1 + n_2 = n$· το πρόσημο του αριθμού θα κωδικοποιείται σαν και τα n ψηφία του αριθμού να παρίσταναν έναν ακέραιο, προσθέτοντας τη μονάδα στο δεξιότερο κλασματικό του ψηφίο. Αυτή είναι η παράσταση *σταθερής υποδιαστολής* (fixed point representation).

Για την παράσταση πραγματικών αριθμών διαθέτουμε 8 ψηφία, και διαθέτουμε 5 ψηφία στο ακέραιο μέρος και 3 ψηφία στο κλασματικό. Η παράσταση των αρνητικών αριθμών γίνεται με το συμπλήρωμα του 2. Ο μεγαλύτερος θετικός αριθμός που μπορούμε να παραστήσουμε έχει την

παράσταση 01111,111 και είναι ο 15,875. Ο μικρότερος αρνητικός αριθμός που μπορούμε να παραστήσουμε είναι ο 10000,000 που έχει την τιμή -16.

Οι πλησιέστεροι αριθμοί στο 0 που μπορούμε να παραστήσουμε (εκτός από το 0 βέβαια, που παριστάνεται ως 00000,000) είναι:

- $00000{,}001_{(2)}$ = 0,125 για τους θετικούς και
- $11111{,}111_{(2)}$ = -0,125 για τους αρνητικούς.

Στην παράσταση σταθερής υποδιαστολής οι πράξεις γίνονται ακριβώς όπως και στους ακέραιους αριθμούς.

Το άθροισμα των αριθμών $01001{,}110_{(2)}$ = 9,75 και $10010{,}001_{(2)}$ = -13,875 υπολογίζεται απευθείας και είναι $11011{,}111_{(2)}$ = -4,125

Το σπουδαιότερο μειονέκτημα της παραστάσεως σταθερής υποδιαστολής είναι ότι το διάστημα των αριθμών που μπορούν να παρασταθούν δεν είναι πολύ μεγάλο.

Παράσταση κινητής υποδιαστολής

Συνήθως στους υπολογιστές χρησιμοποιείται η παράσταση *κινητής υποδιαστολής* (floating point representation).

$$N = \sigma \cdot 2^{\varepsilon}$$

Συντελεστής **(mantissa). Έχει το ίδιο πρόσημο με τον αριθμό N.**

Εκθέτης **(exponent).**

Στην παράσταση κινητής υποδιαστολής ο δυαδικός αριθμός N που θέλουμε να παραστήσουμε εκφράζεται σε *εκθετική μορφή* (exponential representation), σαν ένα γινόμενο δηλαδή ενός κλασματικού αριθμού και μιας δύναμης του 2.

Ο αριθμός $101{,}011_{(2)}$ σε εκθετική μορφή μπορεί να γραφτεί με διάφορες μορφές:

- $0{,}101011 \cdot 2^3$ ⇨ $\sigma = 0{,}101011$, $\varepsilon = 3_{(10)} = 11_{(2)}$
- $1{,}01011 \cdot 2^2$ ⇨ $\sigma = 1{,}01011$, $\varepsilon = 2_{(10)} = 10_{(2)}$
- $10{,}1011 \cdot 2^1$ ⇨ $\sigma = 10{,}1011$, $\varepsilon = 1_{(10)} = 1_{(2)}$

Το ίδιο συμβαίνει και στο δεκαδικό σύστημα: κάθε αριθμός μπορεί να γραφτεί με πολλές εκθετικές μορφές. Το 1023 π.χ. μπορεί να γραφτεί σαν $1{,}023 \cdot 10^3$, σαν $10{,}23 \cdot 10^2$, σαν $0{,}001023 \cdot 10^6$ κλπ.

Βλέπουμε λοιπόν ότι υπάρχουν πολλές εναλλακτικές παραστάσεις ενός αριθμού σε εκθετική μορφή. Στους υπολογιστές έχει επιλεγεί μία από τις παραστάσεις αυτές, η οποία έχει την ιδιότητα $½ \le \sigma < 1$ και ονομάζεται *κανονική μορφή* (normal form).

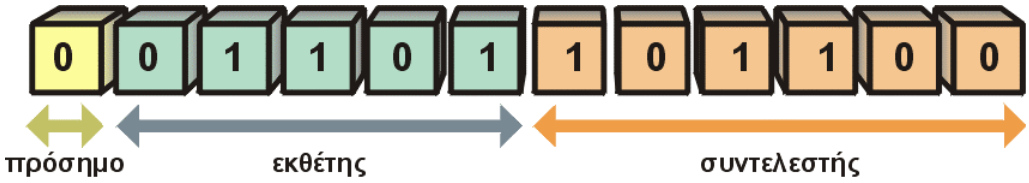
Όταν ο συντελεστής είναι μεταξύ ½ και 1, έχει δύο χαρακτηριστικά:

- Το ακέραιο μέρος του είναι πάντα 0. Έτσι δε χρειάζεται να το αποθηκεύουμε, γιατί η τιμή του είναι γνωστή και δεδομένη.
- Το πρώτο του κλασματικό ψηφίο είναι πάντα 1. Αυτό συμβαίνει, γιατί, οι κλασματικοί αριθμοί που είναι μεγαλύτεροι ή ίσοι από ½ ($=2^{-1}$), στο δυαδικό σύστημα περιέχουν πάντα τον προσθετέο 2^{-1}.

Από όλες τις εκθετικές μορφές του αριθμού $101{,}011_{(2)}$ που είδαμε πιο πριν, η κανονική μορφή είναι η $0{,}101011 \cdot 2^3$.

Η κανονική μορφή του αριθμού $0{,}000100_{(2)}$ είναι η $0{,}100 \cdot 2^{-3}$. Εδώ ο εκθέτης πρέπει να είναι αρνητικός, για να ικανοποιεί ο συντελεστής τη συνθήκη $½ \leq σ < 1$.

Αν λοιπόν όλοι οι αριθμοί είναι εκφρασμένοι στην κανονική εκθετική μορφή, μπορούμε να τους κωδικοποιήσουμε αφιερώνοντας n_1 δυαδικά ψηφία στον εκθέτη και n_2 δυαδικά ψηφία στο συντελεστή, κρατώντας και ένα ψηφίο που θα κωδικοποιεί το πρόσημο του αριθμού, όπως βλέπουμε στο σχήμα.



Το κλασματικό μέρος του συντελεστή παριστάνεται σαν ένας δυαδικός αριθμός με n_1 ψηφία. Εάν ο συντελεστής έχει λιγότερα από n_1 ψηφία, προσθέτουμε μηδενικά στο τέλος, ενώ αν έχει περισσότερα από n_1 ψηφία, τότε τον στρογγυλοποιούμε. Στη *στρογγυλοποίηση* (rounding), αγνοούμε τα ψηφία που περισσεύουν, αλλά, εάν το πρώτο ψηφίο που περισσεύει είναι 1, τότε προσθέτουμε 1 στο λιγότερο σημαντικό ψηφίο του συντελεστή. Ο εκθέτης παριστάνεται και αυτός σαν ένας δυαδικός αριθμός με n_2 ψηφία.

Το πιο σημαντικό από τα ψηφία της λέξης, που παίζει το ρόλο του πρόσημου: έχει την τιμή 0, αν ο αριθμός είναι θετικός και την τιμή 1 αν είναι αρνητικός. Αυτό είναι το πρόσημο του αριθμού· ο εκθέτης, όπως είδαμε, μπορεί να είναι αρνητικός, οπότε έχει το δικό του πρόσημο.

Οι πράξεις με πραγματικούς αριθμούς κινητής υποδιαστολής είναι πιο πολύπλοκες από ό,τι με τους ακεραίους.

Για να προσθέσουμε δύο πραγματικούς αριθμούς κινητής υποδιαστολής, πρέπει πρώτα να τους μετατρέψουμε ώστε να έχουν τον ίδιο εκθέτη. Αν ο ένας αριθμός έχει εκθέτη $ε_1$ και ο άλλος $ε_2$, και ισχύει $ε_1 < ε_2$, τότε αυξάνουμε τον $ε_1$ κατά $ε_2$-$ε_1$ και «ολισθαίνουμε» το συντελεστή του αριθμού αυτού προς τα δεξιά κατά $ε_2$-$ε_1$ ψηφία. Κατά την ολίσθηση αυτή, τα δεξιότερα ψηφία του αριθμού χάνονται, έτσι ο αριθμός μπορεί να μεταβληθεί. Το τελικό αποτέλεσμα λοιπόν μπορεί να μην είναι ακριβές.

Στη συνέχεια προσθέτουμε τους συντελεστές των αριθμών και γράφουμε ξανά το αποτέλεσμα στην κανονική μορφή στρογγυλοποιώντας το συντελεστή. Σε όλες τις μετατροπές όμως το πλήθος των ψηφίων του συντελεστή παραμένει σταθερό.

Στην παράσταση κινητής υποδιαστολής με 8 ψηφία για το συντελεστή και 4 ψηφία για τον εκθέτη, ο αριθμός $x = 16{,}125_{(10)}$ παριστάνεται ως $0{,}10000001 \cdot 2^5$ και ο αριθμός $y = 4{,}3125_{(10)}$ παριστάνεται ως $0{,}1000101 \cdot 2^3$.

Πρώτα μετατρέπουμε τον αριθμό με το μικρότερο εκθέτη, που είναι ο y. Αυξάνουμε τον εκθέτη του κατά 2 και ολισθαίνουμε το συντελεστή του προς τα δεξιά κατά 2 ψηφία.

x = **0010110000001**
y = **0001110001010**

y = **0010100100010**

x+y = **0010110100011**

Στη συνέχεια προσθέτουμε τους δύο συντελεστές. Το άθροισμα δε χρειάζεται κανονικοποίηση, άρα είναι και το τελικό αποτέλεσμα. Η τιμή του είναι 0,10100011 · 2^5, δηλαδή 20,375. Το σωστό αποτέλεσμα της πρόσθεσης όμως είναι 20,4375. Το σφάλμα οφείλεται στη μετατροπή του y ώστε να έχει τον ίδιο εκθέτη με το x.

Οι πράξεις του πολλαπλασιασμού και της διαίρεσης με πραγματικούς αριθμούς κινητής υποδιαστολής είναι πιο εύκολες.

$$(\sigma_1 \cdot 2^{\varepsilon 1}) \cdot (\sigma_2 \cdot 2^{\varepsilon 2}) = (\sigma_1 \cdot \sigma_2) \cdot 2^{\varepsilon 1+\varepsilon 2}$$
$$(\sigma_1 \cdot 2^{\varepsilon 1}) / (\sigma_2 \cdot 2^{\varepsilon 2}) = (\sigma_1 / \sigma_2) \cdot 2^{\varepsilon 1-\varepsilon 2}$$

Για να πολλαπλασιάσουμε δύο αριθμούς προσθέτουμε τους εκθέτες τους και πολλαπλασιάζουμε τους συντελεστές· μετά φέρνουμε πάλι το αποτέλεσμα στην κανονική μορφή. Για να διαιρέσουμε δύο αριθμούς αφαιρούμε τους εκθέτες, διαιρούμε τους συντελεστές και κανονικοποιούμε το αποτέλεσμα.

Όπως είδαμε και στο προηγούμενο παράδειγμα, πολλές φορές η μετατροπή ή η στρογγυλοποίηση που κάνουμε στους αριθμούς κατά την εκτέλεση των πράξεων επιφέρει ένα μικρό σφάλμα στο αποτέλεσμα. Μετά από μία μεγάλη σειρά πράξεων λοιπόν, τα σφάλματα αυτά «συσσωρεύονται», οπότε τα αποτελέσματα μπορεί να μην είναι ικανοποιητικά.

Το πλήθος n_1 των ψηφίων που χρησιμοποιούνται για τον συντελεστή καθορίζουν αντιστοίχως την ακρίβεια παραστάσεως των αριθμών και το πλήθος n_2 των ψηφίων που χρησιμοποιούνται για τον εκθέτη καθορίζουν το εύρος των αριθμών που μπορούμε να παραστήσουμε.

Στην παράσταση κινητής υποδιαστολής με 5 ψηφία για το συντελεστή και 3 ψηφία για τον εκθέτη, οι κοντινότεροι αριθμοί στο 0 που μπορούμε να παραστήσουμε είναι οι:

000000001 = $0{,}03125_{(10)}$ (θετικός)

111111111 = $-0{,}03125_{(10)}$ (αρνητικός)

Οι αριθμοί αυτοί καθορίζουν την ακρίβεια της παράστασης, γιατί οποιοσδήποτε μικρότερος κλασματικός αριθμός θα παρασταθεί σαν 0.

Ο μεγαλύτερος και ο μικρότερος αριθμός που μπορούν να παρασταθούν είναι οι:

011111111 = $0{,}96875 \cdot 2^7 = 124_{(10)}$

100000001 = $-0{,}96875 \cdot 2^7 = -124_{(10)}$

Οι αριθμοί αυτοί καθορίζουν το εύρος της αναπαράστασης.

Βλέπουμε ότι όταν ο εκθέτης ενός αριθμού είναι μεγάλος, τα κλασματικά ψηφία της τιμής του είναι λίγα (π.χ. ο μεγαλύτερος αριθμός εδώ δεν έχει κανένα κλασματικό ψηφίο). Αντίστροφα, όταν ο εκθέτης είναι μικρός, η τιμή του αριθμού έχει πιο πολλά κλασματικά ψηφία, αλλά είναι πιο μικρή.

Παράσταση χαρακτήρων

Εκτός από αριθμούς, θέλουμε να παραστήσουμε στον υπολογιστή και σύμβολα, όπως είναι τα γράμματα, τα σημεία στίξης, τα αριθμητικά ψηφία. Αυτά τα σύμβολα ονομάζονται *χαρακτήρες* (characters). Όπως οι αριθμοί και όλες οι άλλες πληροφορίες, έτσι και οι χαρακτήρες παριστάνονται στους υπολογιστές σαν μία σειρά από δυαδικά ψηφία.

Κάθε χαρακτήρας αντιστοιχίζεται σε μία σειρά ψηφίων που έχει μήκος συνήθως 8 ή 16 bits. Ένα κείμενο παριστάνεται με την ακολουθία των χαρακτήρων που το αποτελούν.

Η παράσταση των αριθμών με δυαδικά ψηφία είναι πολύ απλή και «αυτονόητη». Με τους χαρακτήρες όμως δε συμβαίνει το ίδιο. Η αντιστοίχιση χαρακτήρων με ακολουθίες από δυαδικά ψηφία μπορεί να γίνει με πάρα πολλούς τρόπους· έτσι για να μπορούν οι υπολογιστές να χρησιμοποιούν από κοινού δεδομένα πρέπει να συμφωνούν στην αντιστοίχιση αυτή.

> Αν ένας υπολογιστής Y_1 παριστάνει το γράμμα «Α» με την ακολουθία των δυαδικών ψηφίων 00110100, και ένας υπολογιστής Y_2 παριστάνει με την ακολουθία αυτή το γράμμα «Ε», τότε όταν ο Y_1 στείλει ένα κείμενο στον Y_2, όλα τα «Α» στο κείμενο θα ερμηνεύονται σαν «Ε»!

Μία κοινώς αποδεκτή αντιστοίχιση χαρακτήρων-δυαδικών ακολουθιών ονομάζεται *σύνολο χαρακτήρων* (character set) και συνήθως είναι καθορισμένη από κάποιο διεθνή οργανισμό προτυποποίησης.

Για ευκολία, επειδή μία ακολουθία δυαδικών ψηφίων ερμηνεύεται πολύ εύκολα σαν ένας δυαδικός αριθμός, το σύνολο χαρακτήρων αναφέρεται σαν αντιστοιχία χαρακτήρων με αριθμούς.

> Αν το γράμμα «Α» παριστάνεται με τη δυαδική ακολουθία 01000001, που σαν δυαδικός αριθμός έχει την τιμή 65, λέμε ότι το γράμμα «Α» παριστάνεται με τον αριθμό 65.

Ένα από τα πρώτα σύνολα χαρακτήρων είναι το ASCII (American Standard Code for Information Interchange), το οποίο στην αρχή παρίστανε τους χαρακτήρες με 7 bits και περιλάμβανε μόνο το λατινικό αλφάβητο. Το 8° bit χρησιμοποιούνταν ως *bit ισοτιμίας* (parity bit) που δήλωνε το πλήθος των bits του με την τιμή 1: αν το πλήθος ήταν περιττό, τότε το 8° bit ήταν 1, αλλιώς ήταν 0. Το bit ισοτιμίας χρησίμευε στην ανίχνευση λαθών κατά τη μετάδοση δεδομένων.

> Ο χαρακτήρας $0110101_{(2)}$ έχει άρτιο πλήθος bits με την τιμή 1, έτσι το bit ισοτιμίας του είναι 0, και η τελική του μορφή είναι **00110101**.
>
> Ο χαρακτήρας $1001111_{(2)}$ έχει περιττό πλήθος bits με την τιμή 1, έτσι το bit ισοτιμίας του είναι 1, και η τελική του μορφή είναι **11001111**.

Αργότερα το πρότυπο ASCII επεκτάθηκε στα 8 bits για να περιλάβει και άλλους χαρακτήρες (π.χ. χαρακτήρες με διακριτικά).

Οι πρώτοι 32 χαρακτήρες του προτύπου ASCII, με τιμές 0-31 είναι «μη εκτυπώσιμοι» (non printable) και χρησιμοποιούνται σαν μηνύματα ελέγχου για οθόνες, εκτυπωτές κλπ.

Ο χαρακτήρας 10 δηλώνει την αλλαγή γραμμής σε μία οθόνη τερματικού, ενώ ο χαρακτήρας 12 δηλώνει το τέλος σελίδας σε μία εκτύπωση.

Για την Ελληνική γλώσσα χρησιμοποιούμε συνήθως μία παραλλαγή του ASCII που έχει μήκος 8 bits και έχει προτυποποιηθεί από τον Ελληνικό Οργανισμό Τυποποίησης με το όνομα ΕΛΟΤ-928. Τα γράμματα του Ελληνικού αλφαβήτου παριστάνονται με τιμές που έχουν 1 στο αριστερότερο bit (δηλαδή τιμές μεγαλύτερες από 127).

Ας δούμε τις πιο ενδιαφέρουσες αντιστοιχίσεις χαρακτήρων-αριθμών από το πρότυπο ΕΛΟΤ-928:

33	!	**64**	@	**95**	_	**126**	~	**192**	ΐ	**224**	ΰ
34	"	**65**	A	**96**	`	**161**	΄	**193**	Α	**225**	α
35	#	**66**	B	**97**	a	**162**	Ά	**194**	Β	**226**	β
36	$	**67**	C	**98**	b	**163**	£	**195**	Γ	**227**	γ
37	%	**68**	D	**99**	c	**164**	¤	**196**	Δ	**228**	δ
38	&	**69**	E	**100**	d	**165**	¥	**197**	Ε	**229**	ε
39	'	**70**	F	**101**	e	**166**	¦	**198**	Ζ	**230**	ζ
40	(	**71**	G	**102**	f	**167**	§	**199**	Η	**231**	η
41	)	**72**	H	**103**	g	**168**	¨	**200**	Θ	**232**	θ
42	*	**73**	I	**104**	h	**169**	©	**201**	Ι	**233**	ι
43	+	**74**	J	**105**	i	**171**	«	**202**	Κ	**234**	κ
44	,	**75**	K	**106**	j	**172**	¬	**203**	Λ	**235**	λ
45	-	**76**	L	**107**	k	**173**	-	**204**	Μ	**236**	μ
46	.	**77**	M	**108**	l	**174**	®	**205**	Ν	**237**	ν
47	/	**78**	N	**109**	m	**175**	—	**206**	Ξ	**238**	ξ
48	0	**79**	O	**110**	n	**176**	°	**207**	Ο	**239**	ο
49	1	**80**	P	**111**	o	**177**	±	**208**	Π	**240**	π
50	2	**81**	Q	**112**	p	**178**	²	**209**	Ρ	**241**	ρ
51	3	**82**	R	**113**	q	**179**	³	**211**	Σ	**242**	ς
52	4	**83**	S	**114**	r	**180**	΄	**212**	Τ	**243**	σ
53	5	**84**	T	**115**	s	**181**	µ	**213**	Υ	**244**	τ
54	6	**85**	U	**116**	t	**182**	¶	**214**	Φ	**245**	υ
55	7	**86**	V	**117**	u	**183**	·	**215**	Χ	**246**	φ
56	8	**87**	W	**118**	v	**184**	Έ	**216**	Ψ	**247**	χ
57	9	**88**	X	**119**	w	**185**	Ή	**217**	Ω	**248**	ψ
58	:	**89**	Y	**120**	x	**186**	Ί	**218**	Ϊ	**249**	ω
59	;	**90**	Z	**121**	y	**187**	»	**219**	Ϋ	**250**	ϊ
60	<	**91**	[	**122**	z	**188**	Ό	**220**	ά	**251**	ϋ
61	=	**92**	\	**123**	{	**189**	½	**221**	έ	**252**	ό
62	>	**93**	]	**124**	|	**190**	Ύ	**222**	ή	**253**	ύ
63	?	**94**	^	**125**	}	**191**	Ώ	**223**	ί	**254**	ώ

Το πρότυπο Unicode

Επειδή οι 256 διαφορετικοί χαρακτήρες που διαθέτει το πρότυπο ASCII δεν επαρκούν για να κωδικοποιηθούν όλα τα αλφάβητα, ιδιαίτερα εκείνα που έχουν ιδεογράμματα, έχει αναπτυχθεί και το πρότυπο Unicode, το οποίο παριστάνει ένα χαρακτήρα με μία ακολουθία 16 bits. Έτσι μπορεί να παραστήσει 65536 χαρακτήρες, που είναι αρκετοί για γλώσσες όπως η Κινέζικη και η Ιαπωνική.

Οι 65536 χαρακτήρες του προτύπου Unicode έχουν διαιρεθεί σε «περιοχές», κάθε μία από τις οποίες περιέχει την κωδικοποίηση των χαρακτήρων μίας γλώσσας. Οι πρώτοι 256 χαρακτήρες, για λόγους συμβατότητας, έχουν αποδοθεί στους ίδιους χαρακτήρες με το πρότυπο ASCII.

Οι ελληνικοί χαρακτήρες στο πρότυπο Unicode βρίσκονται στην περιοχή από $880_{(10)}$ έως $975_{(10)}$.

Οι χαρακτήρες της Ταϊλανδέζικης γραφής βρίσκονται στην περιοχή από $3584_{(10)}$ έως $3711_{(10)}$.

Το μεγάλο πλεονέκτημα του προτύπου Unicode, εκτός τού ότι μπορεί να στεγάσει τους χαρακτήρες όλων των συστημάτων γραφής χωρίς υπερκαλύψεις, είναι ότι μπορεί να χρησιμοποιηθεί για να κωδικοποιήσουμε ένα κείμενο που περιέχει πολλές διαφορετικές γλώσσες.

Το τρίγλωσσο κείμενο της συμφωνίας εμπορικής συνεργασίας μεταξύ Ελλάδας, Αρμενίας και Γεωργίας περιέχει τρία διαφορετικά αλφάβητα (τέσσερα αν περιέχει λέξεις ή φράσεις που είναι γραμμένες με το λατινικό αλφάβητο).

Αν το κείμενο παρασταθεί με το πρότυπο Unicode, όλα τα κείμενα θα είναι κωδικοποιημένα με τους σωστούς χαρακτήρες. Αν πρέπει να χρησιμοποιηθεί το πρότυπο ASCII, θα πρέπει να γραφτούν τρία διαφορετικά κείμενα και κάθε ένα να «διαβάζεται» με την έκδοση του προτύπου ASCII που περιλαμβάνει την αντίστοιχη γλώσσα.

Οι χαρακτήρες που θα περιέχονται σε κάθε κείμενο θα έχουν κωδικοποιήσεις με τιμές (που φαίνονται στον πίνακα:

Γλώσσα	Από	Έως
Ελληνική	$880_{(10)}$	$975_{(10)}$
Αρμενική	$1328_{(10)}$	$1423_{(10)}$
Γεωργιανή	$4256_{(10)}$	$4351_{(10)}$

Ανακεφαλαίωση

Οι πράξεις μεταξύ δυαδικών ακεραίων αριθμών γίνονται παρόμοια με τις πράξεις στο δεκαδικό σύστημα. Ο πιο συνηθισμένος τρόπος παράστασης των αρνητικών ακεραίων αριθμών είναι η παράσταση συμπληρώματος του 2, η οποία διευκολύνει πολύ στην εκτέλεση των αριθμητικών πράξεων. Για την παράσταση των πραγματικών αριθμών υπάρχει η παράσταση σταθερής υποδιαστολής, όπου κάθε αριθμός έχει δεδομένο πλήθος κλασματικών ψηφίων, και η παράσταση κινητής υποδιαστολής, όπου οι αριθμοί παριστάνονται σε κανονική εκθετική μορφή. Η δεύτερη είναι πιο ευέλικτη, αν και οι αριθμητικές πράξεις σε αυτή είναι πιο πολύπλοκες, γι' αυτό και γενικά προτιμάται. Οι χαρακτήρες παριστάνονται σαν ακολουθίες από 8 ή 16 δυαδικά ψηφία, σύμφωνα με ένα πρότυπο κωδικοποίησης όπως το ASCII.

Γλωσσάριο όρων

Bit ισοτιμίας	*Parity Bit*
Εκθέτης	*Exponent*
Εκθετική Μορφή	*Exponential Representation*
Κανονική Μορφή	*Normal Form*
Λέξη	*Word*
Μήκος Λέξης	*Word Length*
Παράσταση Σταθερής Υποδιαστολής	*Fixed Point Representation*
Παράσταση Κινητής Υποδιαστολής	*Floating Point Representation*
Σύνολο Χαρακτήρων	*Character Set*
Συντελεστής	*Mantissa*
Στρογγυλοποίηση	*Rounding*
Υπερχείλιση	*Overflow*
Χαρακτήρας	*Character*

Ερωτήσεις

? Ποιο πρέπει να είναι το άθροισμα τριών δυαδικών ψηφίων για να έχουμε κρατούμενο;

? Γιατί η παράσταση συμπληρώματος του 2 είναι τόσο δημοφιλής;

? Πώς παριστάνεται ο ακέραιος αριθμός -17 σε έναν υπολογιστή με μήκος λέξης 16 bits και παράσταση συμπληρώματος του 2 για αρνητικούς αριθμούς;

? Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα για την παράσταση σταθερής υποδιαστολής των πραγματικών αριθμών;

? Πώς προσθέτουμε δύο αριθμούς σε παράσταση κινητής υποδιαστολής;

? Πώς σχετίζεται η ακρίβεια και το εύρος των αριθμών κινητής υποδιαστολής;

? Τι εννοούμε με τους όρους «συμπλήρωμα ως προς 2» και «παράσταση συμπληρώματος ως προς 2»;

? Τι εννοούμε με τον όρο «υπερχείλιση»;

Μάθημα 2.3 Μέθοδοι Συμπίεσης Δεδομένων

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι να περιγράψει τις βασικές τεχνικές συμπίεσης που χρησιμοποιούνται για την μείωση του όγκου των ψηφιακών δεδομένων.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

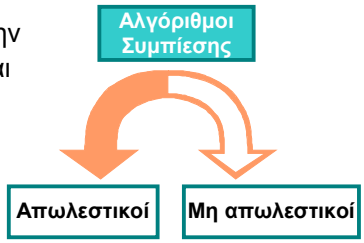
- *Να περιγράφεις τη συμπίεση δεδομένων*
- *Να απαριθμείς τις κατηγορίες συμπίεσης*
- *Να απαριθμείς τους γνωστότερους αλγορίθμους συμπίεσης και να περιγράφεις το πώς λειτουργούν*

Συμπίεση (compression) μιας ακολουθίας δεδομένων, ονομάζουμε την ελάττωση του μεγέθους της ακολουθίας, ώστε να χρειάζεται λιγότερος χώρος για την αποθήκευση ή τη μετάδοσή της. Η διαδικασία της συμπίεσης εφαρμόζεται συστηματικά στα υπολογιστικά συστήματα που χρησιμοποιούν και επεξεργάζονται μεγάλο όγκο ψηφιακών δεδομένων.

Είναι πολύ σημαντικό να γνωρίζουμε ότι η αποτελεσματικότητα μιας μεθόδου ή αλλιώς *αλγορίθμου συμπίεσης* (compression algorithm) εξαρτάται από τη μορφή των δεδομένων στα οποία εφαρμόζεται.

Ένας αλγόριθμος συμπίεσης μπορεί να επιφέρει μεγάλη ελάττωση του μήκους σε ακολουθίες δεδομένων με κάποιο ιδιαίτερο χαρακτηριστικό, ενώ ο ίδιος να είναι αναποτελεσματικός για άλλες ακολουθίες, αφήνοντάς τις στο ίδιο μήκος ή ακόμα και μεγαλώνοντάς το σε μερικές περιπτώσεις.

Όταν έχουμε την ακολουθία των δεδομένων σε συμπιεσμένη μορφή, πρέπει να εφαρμοστεί η αντίστροφη διαδικασία της *αποσυμπίεσης* (decompression, extraction) προκειμένου τα δεδομένα να μπορούν να χρησιμοποιηθούν και πάλι. Η διαδικασία αυτή της αποσυμπίεσης των δεδομένων καθορίζει και τις κατηγορίες των μεθόδων συμπίεσης. Έτσι, υπάρχουν δύο κατηγορίες αλγορίθμων συμπίεσης, οι *απωλεστικοί* (lossy) και οι *μη απωλεστικοί* (lossless) αλγόριθμοι.



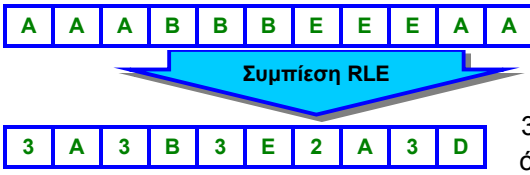
Στους απωλεστικούς αλγορίθμους, όταν γίνει η συμπίεση και μετά ακολουθήσει αποσυμπίεση των δεδομένων, η τελική ακολουθία των δεδομένων διαφέρει από την αρχική. Αντίθετα στους μη απωλεστικούς αλγορίθμους, η διαδικασία συμπίεσης και αποσυμπίεσης επαναφέρει την αρχική ακολουθία. Αν πρέπει να μεταφερθούν δεδομένα με απόλυτη ακρίβεια, πιστότητα, χωρίς να αλλοιωθεί το περιεχόμενό τους, πρέπει να εφαρμοστεί μια μη απωλεστική μέθοδος συμπίεσης. Εφαρμογές ή συστήματα που μεταδίδουν αναλλοίωτες πληροφορίες από το ένα μέσο στο άλλο, για παράδειγμα κάρτες δικτύου ή modem, χρησιμοποιούν τεχνικές μη απωλεστικής συμπίεσης. Υπάρχουν όμως εφαρμογές όπου η μικρή διαφοροποίηση από την αρχική μορφή των δεδομένων δεν επιφέρει σημαντικές αλλαγές. Έτσι, οι περισσότερες εφαρμογές που έχουν να κάνουν με σύνθετες μορφές δεδομένων όπως είναι ο ήχος, η εικόνα, το video, όπου το τελικό αποτέλεσμα αξιολογείται από τον

ανθρώπινο παράγοντα (αυτί, μάτι), μπορούν να κάνουν απωλεστική συμπίεση χωρίς πολλές φορές να υπάρχουν εμφανείς αλλοιώσεις στην ποιότητα των δεδομένων.

Στη συνέχεια θα εξετάσουμε τις αντιπροσωπευτικότερες τεχνικές μη απωλεστικής συμπίεσης.

Αλγόριθμος RLE

Ο *αλγόριθμος RLE* (Run Length Encoding) αποτελεί μια από τις απλούστερες τεχνικές συμπίεσης. Σύμφωνα με τη μέθοδο αυτή, διατρέχεται η ακολουθία των bytes που αποτελούν τα δεδομένα προς συμπίεση και εντοπίζονται οι διαδοχικές επαναλήψεις του ίδιου χαρακτήρα. Στη συνέχεια αντικαθίστανται οι συνεχόμενες επαναλήψεις με το πλήθος τους, ακολουθούμενο από τον χαρακτήρα.



Αν συμπιέσουμε την ακολουθία AAABBBEEEAADDD με τη μέθοδο RLE, τότε η συμπιεσμένη μορφή της ακολουθίας θα είναι 3A3B3E2A3D, που έχει μήκος 10 χαρακτήρες και όχι 14 όπως η αρχική.

Αν προσέξουμε το σχήμα της παραπάνω συμπίεσης θα παρατηρήσουμε ότι αποδίδει ικανοποιητικά για ακολουθίες δεδομένων που έχουν συχνές επαναλήψεις χαρακτήρων. Αν για παράδειγμα συμπιέσουμε την ακολουθία «ABCDE» μήκους 5 bytes με τον αλγόριθμο RLE, η συμπιεσμένη μορφή της είναι «1A1B1C1D1E», με μήκος 10. Στην περίπτωση αυτή όχι μόνο δεν έγινε συμπίεση, αλλά διπλασιάστηκε το μήκος της ακολουθίας. Για να αποφευχθούν αυτές οι περιπτώσεις, ο RLE δε συμπιέζει τις ακολουθίες μη συνεχόμενων χαρακτήρων.

Στην πράξη ο αλγόριθμος διατρέχει την ακολουθία των δεδομένων και αναζητά διαδοχικές επαναλήψεις του ίδιου χαρακτήρα. Όταν βρεθεί μεμονωμένος χαρακτήρας που δεν επαναλαμβάνεται, ή διπλή επανάληψη του, μένει ως έχει, καθώς δεν επωφελούμαστε από την κωδικοποίηση RLE που απαιτεί 2 χαρακτήρες. Όταν όμως βρεθούν διαδοχικές επαναλήψεις ενός χαρακτήρα, τις αντικαθιστά με ένα ζεύγος *(αριθμός, χαρακτήρας)*, όπου «αριθμός» είναι το σύνολο των συνεχόμενων εμφανίσεων του χαρακτήρα.

Αλγόριθμος του Huffman

Αυτή η τεχνική συμπίεσης παρουσιάστηκε από τον D. Huffman το 1952 και επιτυγχάνει μεγάλα ποσοστά συμπίεσης εκμεταλλευόμενη τη στατιστική ανάλυση της ακολουθίας δεδομένων. Η ανάλυση αυτή είναι πιο σύνθετη από την απλή ανίχνευση των συνεχόμενων επαναλήψεων που κάνει ο RLE.

Συμφωνα με την κωδικοποίηση του προτύπου ASCII, σε κάθε χαρακτήρα αντιστοιχεί ένα byte, που είναι ο κωδικός ASCII του χαρακτήρα αυτού. Αν η κωδικοποίηση γίνεται με τον νέο κώδικα Unicode, χρειάζονται 2 bytes για κάθε χαρακτήρα. Το συνολικό μέγεθος της ακολουθίας σε bytes είναι ίσο με το πλήθος των χαρακτήρων επί 1 ή 2 bytes ανά χαρακτήρα (ανάλογα με την κωδικοποίηση).

Στην κωδικοποίηση Huffman, επιτυγχάνεται συμπίεση καθώς ο κώδικας που αντιστοιχεί σε κάθε χαρακτήρα δεν έχει σταθερό μήκος αλλά μεταβλητό. Η κωδικοποίηση για τους πιο συχνά εμφανιζόμενους χαρακτήρες είναι μικρότερη από ό,τι για τους λιγότερο συχνά εμφανιζόμενους, με αποτέλεσμα το συνολικό μέγεθος να είναι μικρότερο, καθώς για τους συχνότερους χρησιμοποιούνται μόνο 2-3 bits και όχι 8 ή 16.

Ο αλγόριθμος συμπίεσης Huffman αποτελείται από τα εξής στάδια:

Βήμα 1: Μετράμε τη συχνότητα του κάθε χαρακτήρα στην ακολουθία

Βήμα 2: Ταξινομούμε τις συχνότητες εμφάνισης σε μια λίστα κατά φθίνουσα τάξη

Βήμα 3: Κατασκευάζουμε ένα «δένδρο» για την κωδικοποίηση ξεκινώντας με τους συχνότερα εμφανιζόμενους χαρακτήρες

Βήμα 4: Αντιστοιχίζουμε τα δυαδικά '0' και '1' σε κάθε κόμβο του δέντρου: Αρχίζοντας από την ρίζα του δέντρου, προσθέτουμε '0' για κάθε αριστερό παιδί και '1' για κάθε δεξί. Οι χαρακτήρες που κωδικοποιούνται είναι τα φύλλα στην βάση του δέντρου. Αρχίζοντας από την κορυφή (ρίζα) του δέντρου, διατρέχοντας το μοναδικό μονοπάτι προς κάθε φύλλο, συλλέγουμε 0 ή 1 και ορίζουμε τον κώδικα για το χαρακτήρα που αντιστοιχεί στο φύλλο αυτό.

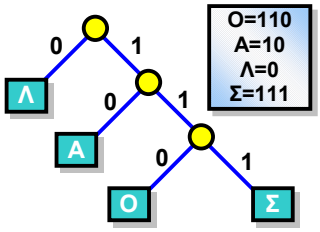
Έστω ότι θέλουμε να κωδικοποιήσουμε τη λέξη «ΑΛΛΟΣ».

Βήμα 1,2: Δημιουργούμε μια λίστα με τους χαρακτήρες ταξινομώντας τους με σειρά εμφάνισης. Έτσι έχουμε τη λίστα {«Λ»,«Α»,«Ο»,«Σ»} αφού το «Λ» εμφανίζεται δύο φορές και οι υπόλοιποι χαρακτήρες από μία (τους οποίους και τοποθετούμε στη λίστα με τη σειρά εμφάνισης τους).

Βήμα 3: Ξεκινώντας από τη ρίζα, δημιουργούμε δύο παιδιά: το «Λ» (το γράμμα που εμφανίζεται πρώτο στη λίστα) και έναν εσωτερικό κόμβο. Στη συνέχεια από τον κόμβο αυτό δημιουργούμε πάλι δύο παιδιά, ένα με τον επόμενο χαρακτήρα στη λίστα, δηλαδή το «Α» και έναν εσωτερικό κόμβο. Διαγράφουμε το «Λ» από τη λίστα. Επαναλαμβάνουμε την διαδικασία μέχρι να τελειώσουν όλοι οι χαρακτήρες της λίστας.

Τελικά όλοι οι χαρακτήρες«Λ», «Α», «Ο», και «Σ» αποθηκεύονται σε φύλλα του δέντρου.

Βήμα 4: Αντιστοιχίζουμε τα δυαδικά 0 και 1, διατρέχοντας το μοναδικό μονοπάτι προς κάθε φύλλο-χαρακτήρα.



Τελικά, η ακολουθία «ΑΛΛΟΣ» που είχε αρχικό μήκος 5×8=40 bits, συμπιεσμένη γίνεται: **1000110111**, συνολικού μήκους 11 bits, άρα περίπου 4 φορές μικρότερη. Βέβαια, στην πράξη, στο τέλος κάθε ακολουθίας προστίθεται και το σχήμα της κωδικοποίησης, ώστε να μπορεί να γίνει η αποσυμπίεση.

Η ακολουθία κωδικοποίησης που παράγεται με τον προηγούμενο αλγόριθμο είναι μοναδική, ώστε να μπορεί να γίνεται η αποσυμπίεση. Έτσι, αν διαθέτουμε το σχήμα της κωδικοποίησης (πίνακας στο πλάι) και την κωδικοποιημένη ακολουθία, τότε μπορούμε να κάνουμε έυκολα την αποσυμπίεση κωδικοποιημένης ακολουθίας **1000110111**. Διαβάζεται από αριστερά προς τα δεξια το πρωτο 1. Δεν υπάρχει χαρακτήρας που να αντιστοιχεί σε αυτό. Διαβάζεται και το 0. Ο χαρακτήρας Α αντιστοιχεί στο 10, άρα αντικαθίσταται το 10 με αυτόν. Στη συνέχεια διαβάζουμε το 0, και ο μοναδικός χαρακτήρας που αρχίζει με 0 είναι ο Λ. Όμοια και για τα υπόλοιπα γράμματα.

Συμπίεση LZW

Πολλά αρχεία, ιδιαίτερα αρχεία χαρακτήρων, έχουν συμβολοσειρές που επαναλαμβάνονται συχνά, για παράδειγμα το άρθρο «τον». Για την αναπαράσταση της λέξης αυτής με χρήση του κώδικα ASCII χρειάζονται 5 bytes, αν συμπεριλάβουμε τα κενά πριν και μετά τη λέξη. Αν

αντιστοιχίσουμε έναν αριθμό σε ολόκληρη τη λέξη (π.χ. το 256 που έχει 2 bytes μήκος), τότε με 2 bytes αντικαθιστούμε παντού τα 5 bytes του άρθρου «τον».

Αυτή είναι η προσέγγιση που ακολουθείται στον αλγόριθμο LZW, ο οποίος παρουσιάστηκε από τους J. Ziv και A. Lempel το 1977, και τροποποιήθηκε από τον T. Welch το 1984.

Ο αλγόριθμος ξεκινά φτιάχνοντας ένα λεξικό όλων των ακολουθιών χαρακτήρων (συμβολοσειρών) που εμφανίζονται στο υπό συμπίεση αρχείο. Στο λεξικό αυτό, που ονομάζεται και *πίνακας αναφορών*, αποθηκεύεται για κάθε συμβολοσειρά ο αριθμός των εμφανίσεων της και δίπλα ένας μοναδικός κωδικός που αντιστοιχεί σε αυτή. Στη φάση της συμπίεσης αντικαθιστούμε κάθε συμβολοσειρά με τον κωδικό της, ο οποίος έχει πολύ μικρότερο μέγεθος από αυτή. Επιπλέον, στο τέλος προσθέτουμε και το λεξικό, ώστε να μπορεί να γίνει η αποσυμπίεση. Για την αποσυμπίεση γίνεται κατ' αρχήν ανάγνωση του λεξικού και αντικατάσταση των κωδικών με τις αντίστοιχες αρχικές συμβολοσειρές. Ο αλγόριθμος LZW χρησιμοποιείται με επιτυχία για αρχεία με μεγάλη κανονικότητα και συχνές επαναλήψεις μεγάλων συμβολοσειρών.

Τα αρχεία τύπου «zip» δημιουργούνται με συμπίεση βασισμένη στον αλγόριθμο LZW.

Ανακεφαλαίωση

Οι αλγόριθμοι συμπίεσης χρησιμοποιούνται για την μείωση του μεγέθους που καταλαμβάνουν τα δεδομένα τα οποία είναι αποθηκευμένα στους υπολογιστές. Οι αλγόριθμοι συμπίεσης διακρίνονται σε απωλεστικούς και μη απωλεστικούς. Οι απωλεστικοί αλγόριθμοι συμπιέζουν πολύ περισσότερο, αλλά όμως είναι αδύνατη η αποσυμπίεση της συμπιεσμένης ακολουθίας στην αρχική μορφή. Αντίθετα, με τους μη απωλεστικούς αλγορίθμους, η αντίστροφη διαδικασία της αποσυμπίεσης μας επαναφέρει στην αρχική ακολουθία των δεδομένων. Οι γνωστότεροι μη απωλεστικοί αλγόριθμοι συμπίεσης είναι ο RLE, ο αλγόριθμος Huffman και ο LZW.

Γλωσσάριο όρων

Αλγόριθμος Huffman	
Αλγόριθμος LZW	
Αλγόριθμος RLE	
Αλγόριθμος Συμπίεσης	*Compression Algorithm*
Αποσυμπίεση	*Decompression-Extraction*
Απωλεστικός Αλγόριθμος	*Lossy Algorithm*
Μη Απωλεστικός Αλγόριθμος	*Lossless Algorithm*
Συμπίεση	*Compression*
Πίνακας Αναφορών	*Reference Table*

Ερωτήσεις

? Τι εννοούμε με τον όρο «συμπίεση δεδομένων»;

? Τι εννοούμε με τον όρο «αποσυμπίεση δεδομένων»;

? Ποιες κατηγορίες αλγορίθμων συμπίεσης δεδομένων υπάρχουν. Περίγραψέ τις.

? Περίγραψε τον αλγόριθμο RLE.

? Περίγραψε τον αλγόριθμο Huffman.

? Περίγραψε τον αλγόριθμο LZW.

Μάθημα 2.4 Ψηφιακή Παράσταση Ήχου, Εικόνας, Video

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να εξηγήσει τις μεθόδους και τα σύγχρονα πρότυπα κωδικοποίησης και ψηφιακής αναπαράστασης σύνθετων μορφών ψηφιακών δεδομένων όπως ήχοι, εικόνες και video.

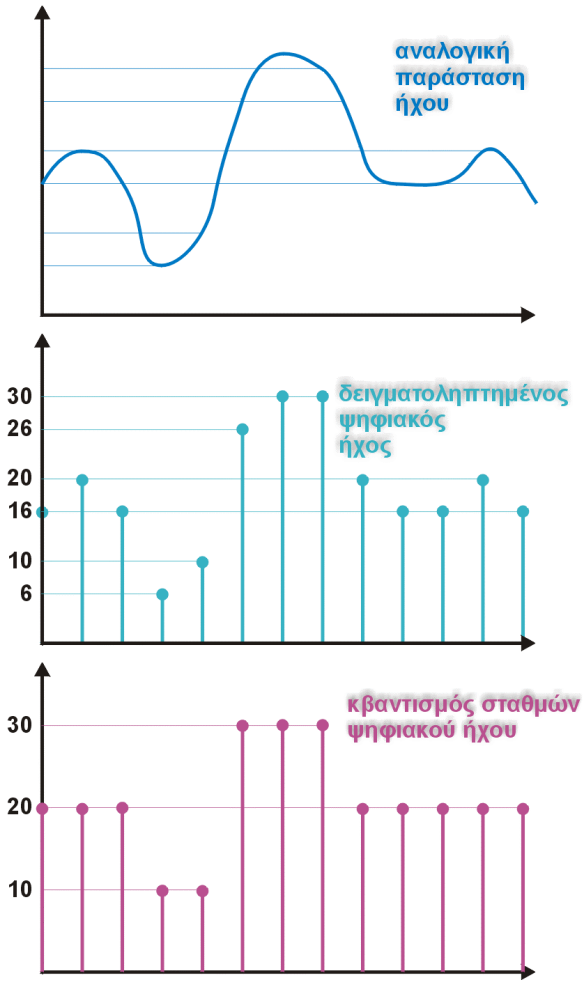
Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις τα πρότυπα κωδικοποίησης ήχου*
- *Να αναφέρεις τις μορφές αποθήκευσης ψηφιακής εικόνας*
- *Να απαριθμείς τι τεχνικές κωδικοποίησης video υπάρχουν*

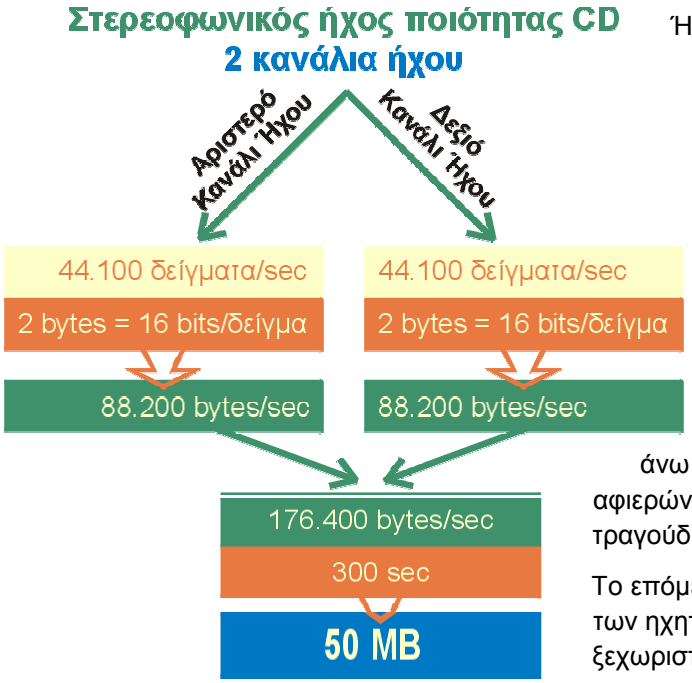
Ψηφιακή Παράσταση Ήχου

Η ψηφιοποίηση του αναλογικού ήχου γίνεται με την περιοδική λήψη δειγμάτων από το αναλογικό σήμα πολλές φορές το δευτερόλεπτο, η οποία λέγεται *δειγματοληψία* (sampling). Ο αριθμός των δειγμάτων που παίρνουμε ανά δευτερόλεπτο, ώστε ο ψηφιακός ήχος να έχει την ίδια ποιότητα με τον αναλογικό, καθορίζεται από τη μέγιστη συχνότητα που εμφανίζει ο αναλογικός ήχος μας. Ο αριθμός αυτός πρέπει να είναι τουλάχιστο ίσος με το διπλάσιο της μέγιστης συχνότητας του ήχου, σύμφωνα με το θεώρημα του Shannon. Το ανθρώπινο αυτί αντιλαμβάνεται ήχους συχνοτήτων από 20Hz έως 20KHz. Έτσι, για να έχουμε πιστή αναπαραγωγή του αναλογικού ήχου χρειάζονται πάνω από 40.000 δείγματα ανά δευτερόλεπτο.



Το πλήθος των δειγμάτων ή αλλιώς ο *ρυθμός* (rate) ή *συχνότητα δειγματοληψίας* (sampling frequency), δεν είναι το μόνο στοιχείο που καθορίζει την κωδικοποίηση του ψηφιακού ήχου. Το κάθε δείγμα αντικατοπτρίζει την ένταση του ήχου για τη στιγμή της δειγματοληψίας στην οποία αντιστοιχεί. Στον αναλογικό ήχο, οποιαδήποτε τιμή έντασης είναι επιτρεπτή. Στον ψηφιακό ήχο όμως, το *πλήθος των bits* (number of bits), που χρησιμοποιούμε για την αποθήκευση του κάθε δείγματος, καθορίζει και τον αριθμό των διαφορετικών τιμών εντάσεως που μπορεί να εμφανιστεί. Συνεπώς, κοντινές, αλλά διαφορετικές τιμές αναλογικής έντασης αντιστοιχούν στην ίδια ψηφιακή τιμή. Το φαινόμενο αυτό λέγεται

κβαντισμός (quantization) των σταθμών έντασης του ήχου. Είναι φανερό ότι όσο πιο πολλά bits χρησιμοποιούμε για την αποθήκευση του κάθε δείγματος, τόσο πιο πιστή αναπαράσταση του αναλογικού ήχου πετυχαίνουμε. Αυτός ο τρόπος κωδικοποίησης λέγεται *παλμοκωδική κωδικοποίηση* (**P**ulse **C**ode **M**odulation).



Ήχος σε ποιότητα CD χρειάζεται δείγματα των 16 bits (2 bytes) και ρυθμό δειγματοληψίας 44,1 KHz (χιλιάδες δείγματα ανά δευτερόλεπτο) για κάθε κανάλι. Έτσι για ψηφιακό στερεοφωνικό ήχο (2 κανάλια ήχου) ποιότητας CD έχουμε 176.400 bytes/δευτερόλεπτο· ένα τραγούδι διάρκειας πέντε λεπτών καταλαμβάνει περίπου 50MB.

Αν οι ανάγκες μας σε ποιότητα ήχου είναι μεγάλες, τότε η παλμοκωδική κωδικοποίηση έχει πολύ μεγάλες αποθηκευτικές απαιτήσεις. Σε συνήθεις εφαρμογές πολυμέσων δεν έχουμε απαιτήσεις στερεοφωνικού ήχου ή ακουστικών συχνοτήτων άνω των 4-5KHz (συχνότητα δειγματοληψίας 11KHz) και αφιερώνουμε 1 byte για κάθε δείγμα. Έτσι για το παραπάνω τραγούδι των 5 λεπτών χρειαζόμαστε το $^{1}/_{16}$ των 50MB.

Το επόμενο βήμα για να μειώσουμε περισσότερο το μέγεθος των ηχητικών δεδομένων είναι να μην κωδικοποιούμε ξεχωριστά το κάθε δείγμα δίνοντας του π.χ. 8 bits. Είναι πιο οικονομικό να βρίσκουμε τη διαφορά του κάθε δείγματος με το προηγούμενο και να κωδικοποιούμε αυτήν. Επειδή ο αριθμός των δειγμάτων ανά δευτερόλεπτο είναι μεγάλος, η πιθανότητα δυο διαδοχικά δείγματα να έχουν κοντινές τιμές έντασης είναι μεγάλη, άρα έχουμε να κωδικοποιήσουμε μικρές διαφορές και δεσμεύουμε λίγα bits, συνήθως 2-3 αντί των 8. Η κωδικοποίηση αυτή λέγεται *διαφορική παλμοκωδική* (**D**ifferential **P**ulse **C**ode **M**odulation). Μια παραλλαγή της μεθόδου αυτής που επιτρέπει την ύπαρξη μεγαλύτερων διαφορών λέγεται *προσαρμοστική-διαφορική παλμοκωδική* (**A**daptive **D**ifferential **P**ulse **C**ode **M**odulation) και χρησιμοποιείται στα αρχεία .wav των Windows.

Όλες αυτές οι τεχνικές αντιμετωπίζουν το ηχητικό σήμα σαν μαθηματική κυματομορφή, αγνοώντας τις ιδιαιτερότητες του ανθρώπινου αυτιού. Έτσι, όλες οι συχνότητες αντιμετωπίζονται ισοδύναμα. Στην πράξη όμως το ανθρώπινο αυτί είναι περισσότερο ευαίσθητο σε κάποια ζώνη συχνοτήτων από κάποια άλλη. Νέες τεχνικές κωδικοποίησης που βασίζονται σε αυτή την απλή παρατήρηση, επιτυγχάνουν πολύ μεγαλύτερα ποσοστά συμπίεσης, χωρίς να υπάρχει ιδιαίτερη απώλεια σε ποιότητα ήχου. Το πρότυπο MPEG-3 (Motion Picture Expert Group) πετυχαίνει συμπίεση έως και 12 φορές ως προς την αρχική μορφή χωρίς εμφανείς απώλειες στην ποιότητα. Τα αρχεία .mp3 είναι κωδικοποιημένα με το πρότυπο αυτό. Αν όμως δεχθούμε συμβιβασμούς στην ακουστική ποιότητα, π.χ. για μονοφωνικό ήχο ποιότητας τηλεφωνικής μετάδοσης, τότε επιτυγχάνεται συμπίεση μέχρι και 100 φορές.

Ψηφιακή Παράσταση Εικόνας

Μία ψηφιακή εικόνα αποτελείται από μικρές κουκίδες που ονομάζονται *εικονοστοιχεία* (pixels). Στην πιο απλή περίπτωση κάθε εικονοστοιχείο μπορεί να είναι είτε άσπρο είτε μαύρο.

Στο διπλανό σχήμα βλέπουμε μία ασπρόμαυρη ψηφιακή εικόνα. Η εικόνα του σχήματος αποτελείται από 256 γραμμές και 256 στήλες, περιέχει δηλαδή 65.536 διαφορετικά εικονοστοιχεία. Επειδή η φωτογραφία είναι ασπρόμαυρη, κάθε εικονοστοιχείο κωδικοποιείται με ένα bit: το 0 συμβολίζει το μαύρο χρώμα και το 1 το άσπρο. Έτσι, για την αποθήκευση αυτής της φωτογραφίας στον υπολογιστή χρειαζόμαστε 65.536 bits ή ισοδύναμα 8.192 bytes.

Για έγχρωμες εικόνες, κάθε εικονοστοιχείο απαιτεί περισσότερα από 1 bit για την αναπαράσταση του αντίστοιχου χρώματος. Για παράδειγμα, αν έχουμε 16 διαφορετικά χρώματα και κάθε χρώμα εμφανίζει 16 αποχρώσεις, τότε χρειαζόμαστε 16×16=256 διαφορετικές τιμές. Όλες αυτές μπορούμε να τις αναπαραστήσουμε με ένα byte (8 bits). Αν, για παράδειγμα, έχουμε μια εικόνα μεγέθους 640×480, δηλαδή 307.200 εικονοστοιχεία και για καθένα χρησιμοποιούμε 3 bytes=24bit για την αναπαράσταση του χρώματος, τότε χρειαζόμαστε 921.600 bytes. Τα αρχεία που περιέχουν την εικόνα ασυμπίεστη, ονομάζονται *χαρτογραφικά* (bitmap, BMP).

Το μέγεθος των ασυμπίεστων εικόνων αυξάνει δραματικά με την αύξηση των διαστάσεων της ή του βάθους χρώματος (δηλ. τον αριθμό των bytes για την αναπαράσταση των διαφορετικών χρωμάτων). Η χρησιμοποίηση της μεθόδου συμπίεσης RLE σε εικόνες που είναι σκίτσα ή σχεδιαγράμματα, κυρίως με αποχρώσεις του γκρι, οδηγεί σε σημαντική μείωση του όγκου τους. Η RLE μέθοδος είναι πολύ αποτελεσματική για εικόνες με συνεχόμενες περιοχές εικονοστοιχείων που έχουν το ίδιο χρώμα, καθώς τα αντικαθιστά με ένα κωδικό και το πλήθος τους (βλέπε κεφάλαιο 2.3). Τα αρχεία αυτά είναι τα PCX αλλά και πολλές μορφές TIFF.

Η συμπίεση LZW εφαρμόζεται σε περισσότερο σύνθετες εικόνες, όπως π.χ. έγχρωμες φωτογραφίες. Εκεί, υπάρχει μεγάλη ποικιλία χρωμάτων, που όμως επαναλαμβάνονται πολλές φορές τα ίδια, άρα, όπως είδαμε στο κεφάλαιο 2.3, η κωδικοποίηση τους με λιγότερα bit φέρνει μεγάλα ποσοστά συμπίεσης. Τέτοια συμπίεση υπάρχει στα αρχεία GIF.

Τέλος, αντίθετα με τις προηγούμενες μεθόδους αναπαράστασης που χρησιμοποιούσαν μη απωλεστικούς αλγόριθμους συμπίεσης, υπάρχει μια ευρύτατα διαδεδομένη μέθοδος για εικόνες, που επιτυγχάνει συμπίεση μέχρι και 100-200 φορές σε ορισμένες περιπτώσεις, με μικρή απώλεια της ποιότητας. Η κωδικοποίηση αυτή λέγεται κωδικοποίηση JPEG (Joint Photographic Expert Group). Βασίζεται στην εφαρμογή μιας σειράς πολύπλοκων μαθηματικών μετασχηματισμών στην εικόνα. Για τη συμπίεση ή την αποσυμπίεση από JPEG χρειάζεται αρκετή υπολογιστική ισχύς, κάτι που παλαιότερα ήταν ιδιαίτερα χρονοβόρα. Σήμερα, που υπάρχουν ισχυροί επεξεργαστές, η κωδικοποίηση JPEG δεν είναι πια πρόβλημα, και έχει το πλεονέκτημα της δυνατότητας επιλογής του ποσοστού συμπίεσης που θέλουμε να επιτύχουμε, με αντίστοιχη επιβάρυνση στην ποιότητα της εικόνας. Συνήθως συμπιέσεις 10 έως και 20 φορές, οδηγούν σε εικόνες με μη ορατή διαφορά από την αρχική.

Αντίθετα με την παραπάνω μορφή ψηφιακής αναπαράστασης εικόνας όπου υπάρχει κωδικοποίηση της κάθε εικονοστοιχείου με συγκεκριμένο αριθμό bits, η διανυσματική αναπαράσταση βασίζεται στα γεωμετρικά χαρακτηριστικά των σχημάτων που απεικονίζουν την εικόνα. Έτσι αν π.χ. είχαμε να αναπαρταστήσουμε ένα τρίγωνο, τότε η διανυσματική αναπαράσταση περιλαμβάνει την αποθήκευση των καρτεσιανών συντεταγμένων των τριών κορυφών. Κάθε κορυφή θεωρείται εικονοστοιχείο και αποθηκεύτεται όπως πριν. Όμως τώρα έχουμε τρία σημεία που ορίζουν όλο το σχήμα. Η διανυσματική μορφή παράγεται κυρίως από προγράμματα σχεδίασης (CAD – Computer Aided Design). Ο υπολογιστής σχεδιάζει τα γεωμετρικά σχήματα με βάση τις συντεταγμένες που διαβάζει από τέτοια αρχεία

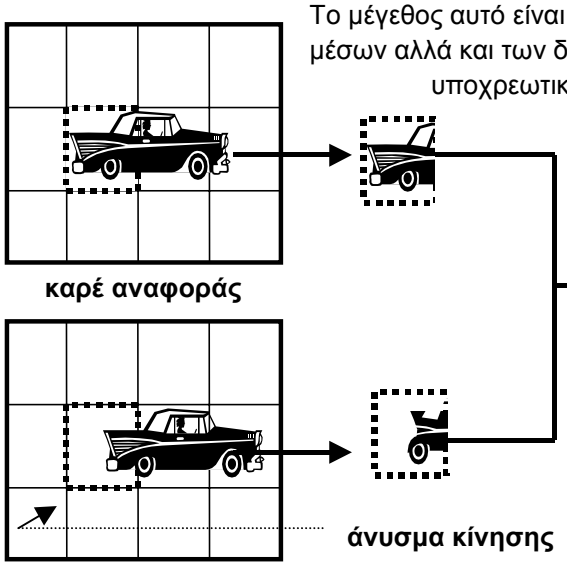
Ψηφιακή Παράσταση Video

Τα τελευταία χρόνια η εξέλιξη στις μεθόδους συμπίεσης έχει επιτρέψει την αναπαραγωγή κινούμενης εικόνας ή video μέσω των υπολογιστών χαμηλού κόστους. Η κινούμενη εικόνα είναι πιο σύνθετη από την απλή εικόνα ή τον ήχο, καθώς περιέχει εικόνα και ήχο που συγχρονίζονται. Η αρχή πάνω στην οποία στηρίζεται η αναπαραγωγή video είναι ίδια με αυτή του κινηματογράφου. Σε κάθε δευτερόλεπτο εναλλάσσονται αρκετές εικόνες με ταχύτητα και μικροδιαφορές μεταξύ τους, δίνοντας στον άνθρωπο την ψευδαίσθηση της κίνησης. Το ανθρώπινο μάτι δεν μπορεί να αντιληφθεί εύκολα αλλαγές με ρυθμό πάνω από 20 εικόνες/δευτερόλεπτο και επομένως νομίζει πως βλέπει μια συνεχόμενη σκηνή. Στον κινηματογράφο έχουμε 24 εικόνες (ή καρέ όπως λέγονται) ανά δευτερόλεπτο, ενώ στην τηλεόραση κυμαίνονται από 25 καρέ/δευτερόλεπτο (σύστημα PAL στην Ευρώπη) εως και 30 καρέ/δευτερόλεπτο (NTSC στην Αμερική).

Οι συχνότητες αυτές αναφέρονται στο **ρυθμό εναλλαγής των ακίνητων εικόνων** για να σχηματιστεί το video και δεν έχουν σχέση με τη συχνότητα ανανέωσης του πλαισίου που θα δούμε στην τεχνολογία των οθονών.

Αν μια εικόνα διαστάσεων 640×480 εικονοστοιχείων με 24 bits χρώματος για κάθε εικονοστοιχείο, έχει κανονικό μέγεθος 920 KB, τότε για ένα δευτερόλεπτο video με ρυθμό 25 εικόνων ανά δευτερόλεπτο θα χρειαζόμασταν 24MB περίπου. Για μια ταινία 90 λεπτών θα χρειαζόμασταν 24MB/sec × 60sec/min × 90min περίπου 3500 GB!!

Το μέγεθος αυτό είναι τεράστιο για την σημερινή τεχνολογία των αποθηκευτικών μέσων αλλά και των δικτύων μετάδοσης δεδομένων. Η συμπίεση εδώ είναι υποχρεωτική.



Για να συμπιεστεί το video, η κάθε ακίνητη εικόνα συμπιέζεται με κάποιο απωλεστικό αλγόριθμο συμπίεσης εικόνας, συνήθως τον JPEG. Αυτό όμως δεν αρκεί καθώς έστω και υποδεκαπλασιασμός του μεγέθους της ακίνητης εικόνας, θα είχε στο παραπάνω παράδειγμα αντίστοιχη μείωση στην τιμή των 350GB που εξακολουθεί να είναι τεράστιο.

Ο μεγάλος όγκος δεδομένων προέρχεται από τον μεγάλο αριθμό των εικόνων ή πλαισίων, όπως θα τα

λέμε στο εξής, ανά δευτερόλεπτο. Η αίσθηση της κίνησης βασίζεται στην ταχύτατη εναλλαγή εικόνων που διαφέρουν πολύ λίγο μεταξύ τους, ώστε το μάτι να μην βλέπει ξαφνικά μεγάλες διαφορές. Η τεχνολογία συμπίεσης σε κινούμενη εικόνα βασίζεται στην απομάκρυνση της πλεονάζουσας πληροφορίας που επαναλαμβάνεται στα διαδοχικά πλαίσια. Η τεχνική που χρησιμοποιείται για την ανίχνευση της πλεονάζουσας πληροφορίας, βασίζεται στην σύγκριση διαδοχικών πλαισίων και στην εύρεση των διαφορών μεταξύ τους. Στη συνέχεια αντί να κωδικοποιηθεί το πλαίσιο, κωδικοποιούμε μόνο τις διαφορές. Το τρέχον πλαίσιο ανακατασκευάζεται από το προηγούμενο και τη διαφορά του με αυτό.

Στην πράξη τα σχήματα κωδικοποίησης είναι αρκετά πιο σύνθετα από την παραπάνω περιγραφή. Η διαδικασία εύρεσης των διαφορών λέγεται *αλγόριθμος εκτίμησης κίνησης* (motion estimation algorithm). Έχουμε δύο διαδοχικά πλαίσια που ονομάζουμε προηγούμενο και τρέχον. Τα δύο πλαίσια διαιρούνται σε στοιχειώδεις μικρές τετραγωνικές περιοχές (blocks). Η κάθε περιοχή του τρέχοντος πλαισίου αναζητείται στο προηγούμενο. Όταν βρεθεί, αντί να κωδικοποιήσουμε την περιοχή με κάποιο αλγόριθμο κωδικοποίησης εικόνας, κωδικοποιούμε το διάνυσμα που δείχνει τις συντεταγμένες του σημείου μετατόπισης, άρα σημαντικά λιγότερη πληροφορία και επομένως επιτυγχάνουμε μεγάλη συμπίεση.

Το πιο διαδεδομένο πρότυπο συμπίεσης για κινούμενη εικόνα είναι το MPEG-2. Για να κωδικοποιήσουμε ένα video σύμφωνα με το πρότυπο αυτό, διαιρούμε το συνολικό αριθμό των 25 ή 30 ή περισσοτέρων πλαισίων το δευτερόλεπτο σε κατηγορίες.

Ανακεφαλαίωση

Η ψηφιακή αναπαράσταση σύνθετων μορφών δεδομένων που χρησιμοποιούνται κυρίως σε εφαρμογές πολυμέσων γίνεται με την κωδικοποίησή τους σε μορφές που προσφέρουν αρκετή συμπίεση, ώστε να ελαττωθεί το μεγάλο τους μέγεθος. Ο αναλογικός ήχος υπόκειται σε δειγματοληψία και κβαντοποιείται σε ψηφιακές τιμές. Ο ψηφιακός πλέον ήχος κωδικοποιείται με μεθόδους όπως η παλμοκωδική ή η διαφορική παλμοκωδική ή η μέθοδος MPEG-Layer III (.mp3).

Η ψηφιακή αναπαράσταση της εικόνας βασίζεται στην ανάλυσή της σε εικονοστοιχεία. Το κάθε εικονοστοιχείο καταλαμβάνει χώρο ανάλογο του πλήθους των χρωμάτων που χρησιμοποιούνται για την εικόνα. Υπάρχουν τεχνικές κωδικοποίησης που οδηγούν σε απωλεστική συμπίεση (π.χ. JPEG), χωρίς όμως ορατές διαφορές στην εμφάνιση της εικόνας.

Το video αποτελείται από μια σειρά εικόνων που εναλλάσσονται γρήγορα στο χρόνο, ώστε να δημιουργούν την ψευδαίσθηση της κίνησης. Η αναπαράσταση της κινούμενης εικόνας βασίζεται στην κωδικοποίηση των διαφορετικών πλαισίων-εικόνων που αποτελούν το video με τεχνικές αναπαράστασης εικόνων και στην απαλοιφή των κοινών σημείων μεταξύ διαδοχικών πλαισίων, που επαναλαμβάνονται, σχεδόν τα ίδια, πολλές φορές το δευτερόλεπτο.

Γλωσσάριο όρων

Αλγόριθμος Εκτίμησης Κίνησης	*Motion Estimation Algorithm*
Δειγματοληψία	*Sampling*
Διαφορική Παλμοκωδική Κωδικοποίηση	*Differential Pulse Code Modulation*
Εικόνα ή Πλαίσιο ή Καρέ	*Frame*
Κβαντισμός	*Quantization*
Κωδικοποιήσεις	*BMP- Bitmap* *GIF - Graphics Interchange Format* *JPEG - Joint Picture Expert Group* *MPEG - Motion Picture Expert Group*
Παλμοκωδική Κωδικοποίηση	*Pulse Code Modulation*
Προσαρμοστική - Διαφορική Παλμοκωδική Κωδικοποίηση	*Adaptive Differential Pulse Code Modulation*
Ρυθμός	*Rate*
Συχνότητα Δειγματοληψίας	*Sampling Frequency*
Εικονοστοιχείο	*Pixel*

Ερωτήσεις

? Τι λέγεται δειγματοληψία;

? Από τι εξαρτάται η ποιότητα του ψηφιακού ήχου;

? Τι εννοούμε με τον όρο «κβαντισμό»;

? Τι λέγεται παλμοκωδική κωδικοποίηση;

? Τι λέγεται διαφορική παλμοκωδική κωδικοποίηση;

? Πώς παριστάνεται ο ήχος ψηφιακά;

? Πώς παριστάνεται η κινούμενη εικόνα;

Τι μάθαμε σε αυτό το κεφάλαιο

- Όλα τα δεδομένα παριστάνονται στον υπολογιστή σαν ακολουθίες από 0 και 1.
- Ένα αριθμητικό σύστημα με βάση β παριστάνει κάθε αριθμό ως άθροισμα δυνάμεων του αριθμού β..
- Ο πιο συνηθισμένος τρόπος παράστασης των αρνητικών ακεραίων αριθμών είναι η παράσταση συμπληρώματος του 2. Για την παράσταση των πραγματικών αριθμών χρησιμοποιείται συνήθως η παράσταση κινητής υποδιαστολής
- Οι χαρακτήρες παριστάνονται σαν ακολουθίες από 8 ή 16 δυαδικά ψηφία, σύμφωνα με ένα πρότυπο κωδικοποίησης.
- Οι αλγόριθμοι συμπίεσης χρησιμοποιούνται για την μείωση του μεγέθους που καταλαμβάνουν τα δεδομένα τα οποία είναι αποθηκευμένα στους υπολογιστές.
- Ο αναλογικός ήχος δειγματοληπτείται και κβαντοποιείται σε ψηφιακές τιμές.
- Η ψηφιακή αναπαράσταση της εικόνας βασίζεται στην ανάλυσή της σε εικονοστοιχεία.
- Το video αποτελείται από μια σειρά εικόνων που εναλλάσσονται γρήγορα στο χρόνο, ώστε να δημιουργούν την ψευδαίσθηση της κίνησης.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Π. Τσανάκα, Γ. Φραγκάκη, Αρχιτεκτονική Υπολογιστών, Συμμετρία, 1987.
- Hayes J., Computer Architecture and Organization, McGraw-Hill, 1988.
- Patterson D., Hennessy J. Computer Organization & Design, The Hardware/Software Interface, Morgan Kaufmann, 1994.

Κεφάλαιο

Οργάνωση και Λειτουργία Επεξεργαστών

Σκοπός του κεφαλαίου αυτού είναι να περιγράψει την εσωτερική οργάνωση των υπολογιστών, ιδιαίτερα των επεξεργαστών, και τον τρόπο προγραμματισμού τους, σε γλώσσα μηχανής και σε συμβολική γλώσσα.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις την οργάνωση ενός υπολογιστή και τις βασικές μονάδες που τον αποτελούν.
- Να διακρίνεις τις μονάδες οι οποίες αποτελούν τον επεξεργαστή.
- Να γράφεις απλά προγράμματα σε γλώσσα μηχανής ενός εκπαιδευτικού υπολογιστή και να εξηγείς την εκτέλεσή τους.
- Να γράφεις προγράμματα σε συμβολική γλώσσα για τον ίδιο εκπαιδευτικό υπολογιστή.
- Να εξηγείς πώς εκτελεί ο υπολογιστής τα προγράμματα σε διάφορες γλώσσες προγραμματισμού.
- Να απαριθμείς τις διάφορες σύγχρονες αρχιτεκτονικές επεξεργαστών.

Μαθήματα

3.1 Τμήματα του Υπολογιστή

3.2 Γλώσσα Μηχανής

3.3 Συμβολική Γλώσσα

3.4 Από τη Γλώσσα Μηχανής στα Προγράμματα Εφαρμογής

3.5 Σύγχρονοι Επεξεργαστές

Μάθημα 3.1 Τμήματα του Υπολογιστή

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει τη βασική εσωτερική δομή ενός υπολογιστή και ιδιαίτερα της Κεντρικής Μονάδας Επεξεργασίας· να δείξει τη χρησιμότητα και τη λειτουργία κάθε μονάδας, και τις αλληλεπιδράσεις μεταξύ τους.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

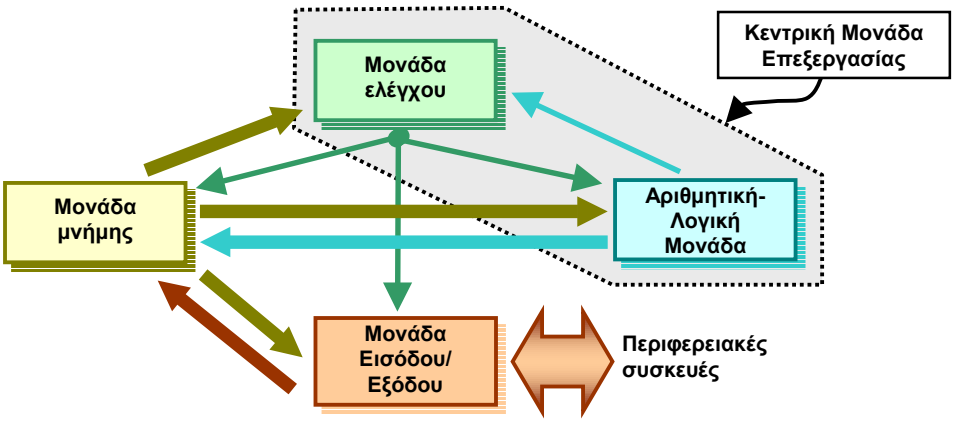
- *Να απαριθμείς τα βασικά τμήματα ενός υπολογιστή*
- *Να περιγράφεις τη λειτουργία κάθε μίας από τις βασικές μονάδες του υπολογιστή*
- *Να εξηγείς τον τρόπο που οι μονάδες αλληλεπιδρούν μεταξύ τους*

Κάθε ηλεκτρονικός υπολογιστής αποτελείται από τουλάχιστον τέσσερα κύρια τμήματα:

- ⇨ την *αριθμητική-λογική μονάδα* (arithmetic logical unit - ALU)
- ⇨ τη *μονάδα μνήμης* (memory unit)
- ⇨ τη *μονάδα εισόδου-εξόδου* (Input-Output ή I/O unit)
- ⇨ τη *μονάδα ελέγχου* (control unit).

Η μονάδα ελέγχου μαζί με την αριθμητική-λογική μονάδα αποτελούν την *κεντρική μονάδα επεξεργασίας* ή *ΚΜΕ* (Central Processing Unit - CPU) ή *επεξεργαστή* (processor).

Μολονότι κάθε μία από αυτές τις μονάδες διαφέρει από υπολογιστή σε υπολογιστή ως προς την πολυπλοκότητα, το μέγεθος και το σχεδιασμό της, ωστόσο επιτελεί την ίδια εργασία.



Καταχωρητές

Η ΚΜΕ χρειάζεται κάποιο τρόπο να αποθηκεύει προσωρινά κάποια δεδομένα, τα οποία χρησιμοποιεί κατά τη λειτουργία της.

Για να εκτελέσει μια πρόσθεση, η ΚΜΕ πρέπει να έχει αποθηκεύσει κάπου τους προσθετέους και μετά να καταχωρήσει το αποτέλεσμα.

Έτσι στο εσωτερικό της η ΚΜΕ διαθέτει μικρές μονάδες αποθήκευσης δεδομένων, που έχουν μέγεθος ένα byte ή όσο το μήκος λέξης του υπολογιστή και ονομάζονται *καταχωρητές* (registers).

Οργάνωση του υπολογιστή

Η **αριθμητική-λογική μονάδα** έχει τις ακόλουθες δυνατότητες:

- εκτελεί τις βασικές αριθμητικές πράξεις, δηλαδή την πρόσθεση, αφαίρεση, πολλαπλασιασμό και διαίρεση
- εκτελεί λογικές πράξεις, π.χ. *λογικό αθροισμα* (OR), *λογικό γινόμενο* (AND).
- εκτελεί διάφορες βοηθητικές εργασίες, π.χ. *ολίσθηση* (shift) μίας τιμής.

Οι πράξεις αυτές μπορούν να γίνουν μέσα σε κλάσματα του δευτερολέπτου. Οι μεγάλες όμως ταχύτητες θα ήταν άχρηστες, αν τα δεδομένα για την εκτέλεση των πράξεων έπρεπε να διοχετευθούν στον υπολογιστή από τον άνθρωπο αμέσως πριν από την εκτέλεση κάθε πράξης.

Έτσι οι πληροφορίες για τους υπολογισμούς που πρόκειται να εκτελέσει ο υπολογιστής, δίνονται όλες μαζί στην αρχή και στη συνέχεια αυτός λειτουργεί αυτόματα, χωρίς τη μεσολάβηση του ανθρώπου. Αναγκαία προϋπόθεση για να γίνει αυτό δυνατό είναι η ύπαρξη της μονάδας μνήμης.

Η **μονάδα μνήμης** χρησιμεύει για την αποθήκευση των αρχικών δεδομένων που είναι απαραίτητα για τους υπολογισμούς, των αποτελεσμάτων που προκύπτουν ενδιάμεσα, των τελικών αποτελεσμάτων καθώς και ενός συνόλου εντολών (οδηγιών), τις οποίες ο υπολογιστής πρέπει να εκτελέσει για να πραγματοποιήσει τον υπολογισμό.

Τα αρχικά δεδομένα ονομάζονται *δεδομένα εισόδου* (input data), τα τελικά αποτελέσματα *δεδομένα εξόδου* (output data) και το σύνολο των εντολών *πρόγραμμα* (program).

Η **μονάδα εισόδου/εξόδου** επιτρέπει την επικοινωνία του εξωτερικού περιβάλλοντος με τον υπολογιστή. Στη μονάδα αυτή συνδέονται οι διάφορες περιφερειακές μονάδες, όπως π.χ. ο εκτυπωτής και το πληκτρολόγιο.

Η **μονάδα ελέγχου** έχει σκοπό τον έλεγχο και το συντονισμό όλων των λειτουργιών του υπολογιστή. Παίρνει από τη μνήμη μία προς μία τις εντολές του προγράμματος, τις αναλύει σε στοιχειώδεις εργασίες και στέλνει στις διάφορες μονάδες λεπτομερείς οδηγίες για το ποια λειτουργία πρέπει να εκτελέσουν και πότε.

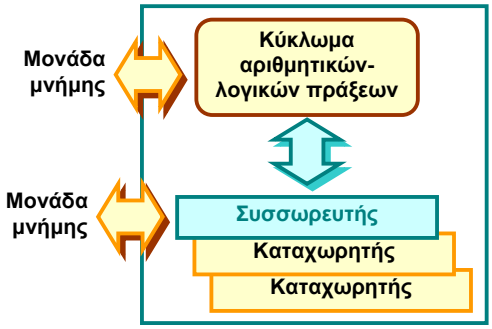
Το μοντέλο αυτό του υπολογιστή δόθηκε για πρώτη φορά από τον Von Neumann το 1947. Σήμερα οι περισσότεροι υπολογιστές ακολουθούν βασικά αυτό το μοντέλο.

Αριθμητική-λογική μονάδα

Η αριθμητική-λογική μονάδα, όπως είδαμε, εκτελεί αριθμητικές και λογικές πράξεις καθώς και διάφορες βοηθητικές εργασίες. Αποτελείται από δύο τμήματα:

- ☑ καταχωρητές που χρησιμεύουν για την προσωρινή αποθήκευση αριθμών
- ☑ κυκλώματα για την εκτέλεση πράξεων και άλλων βοηθητικών εργασιών

Συνήθως σε μία αριθμητική-λογική μονάδα υπάρχει ένας καταχωρητής που χρησιμοποιείται για την αποθήκευση του αποτελέσματος από κάθε πράξη που εκτελεί η μονάδα. Ο καταχωρητής αυτός ονομάζεται *συσσωρευτής* (accumulator), γιατί «συσσωρεύει» τα διάφορα αποτελέσματα των υπολογισμών.



Αριθμητική-Λογική Μονάδα

> Κατά τον υπολογισμό της αριθμητικής έκφρασης ((x+1)/5)*3 + y ο συσσωρευτής θα περιέχει κατά σειρά τις τιμές x, x+1, (x+1)/5, ((x+1)/5)*3, ((x+1)/5)*3 + y.

Για να εκτελεστούν οι πράξεις από την αριθμητική-λογική μονάδα πρέπει πρώτα να μεταφερθούν τα δεδομένα των πράξεων στους καταχωρητές της από τη μνήμη και μετά να μεταφερθούν τα αποτελέσματα από την αριθμητική-λογική μονάδα προς τη μνήμη.

Μονάδα μνήμης

Η μονάδα μνήμης του υπολογιστή αποτελείται από ολοκληρωμένα κυκλώματα που αποθηκεύουν δυαδικά ψηφία, δηλαδή πληροφορίες με τη μορφή 0 ή 1.

Διαιρείται σε *θέσεις μνήμης* (memory positions) και κάθε μια θέση μνήμης μπορεί να αποθηκεύσει μία *λέξη* (word), δηλαδή ένα σύνολο δυαδικών ψηφίων. Το πλήθος των δυαδικών ψηφίων της λέξης ονομάζεται *μήκος λέξης* (word length) και είναι σταθερό για κάθε υπολογιστή· συνήθως είναι μία δύναμη του 2.

Το πλήθος των λέξεων που αποτελούν τη μνήμη είναι το *μέγεθός* της. Όπως το μήκος λέξης, έτσι και το μέγεθος της μνήμης συνήθως είναι μία δύναμη του 2.

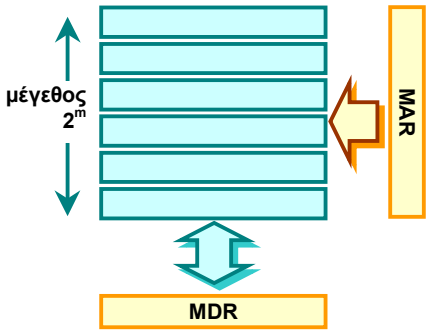
Για να μπορούμε να αναφερθούμε στις θέσεις της μνήμης, τις αριθμούμε. Κάθε θέση μνήμης έτσι έχει ένα αριθμό που αντιστοιχεί σε αυτή και ονομάζεται *διεύθυνση* (address). Αν η διεύθυνση παριστάνεται με m δυαδικά ψηφία, οι διευθύνσεις της μνήμης ξεκινούν από το 0 και φθάνουν έως 2^m-1.

> Σε ένα υπολογιστή που η μνήμη έχει μέγεθος 2^{10} λέξεις, οι διευθύνσεις της μνήμης ξεκινούν από 0 και φθάνουν μέχρι τη 2^{10}-1 = 1023. Οι αριθμοί αυτοί μπορούν να παρασταθούν στο δυαδικό σύστημα με 10 bits, έτσι οι διευθύνσεις στον υπολογιστή αυτό παριστάνονται σαν ένας 10-ψήφιος δυαδικός αριθμός.

Εκτός από τα κυκλώματα που αποθηκεύουν δεδομένα, η μονάδα μνήμης περιλαμβάνει επίσης τουλάχιστον δύο καταχωρητές:

MDR Ο *καταχωρητής δεδομένων της μνήμης* (memory data register - MDR) χρησιμοποιείται για την προσωρινή αποθήκευση των δεδομένων που διαβάζονται ή γράφονται στη μνήμη. Επειδή χρησιμοποιείται για τη μεταφορά δεδομένων, ο MDR έχει ίδιο μέγεθος με τη λέξη του υπολογιστή.

MAR Ο *καταχωρητής διευθύνσεων της μνήμης* (memory address register - MAR) κρατά τη διεύθυνση της λέξης που πρόκειται να διαβαστεί από τη μνήμη ή να γραφεί σε αυτή. Επειδή χρησιμοποιείται για την αποθήκευση διευθύνσεων, το μέγεθος του MAR είναι τέτοιο ώστε να κωδικοποιήσει οποιαδήποτε διεύθυνση.



Για να αποθηκευθεί μία λέξη στη μνήμη, η τιμή της γράφεται στον MDR και η διεύθυνση όπου θα αποθηκευθεί η λέξη γράφεται στον MAR. Μετά ενεργοποιείται η

μονάδα της μνήμης και γίνεται η μεταφορά των δεδομένων στη θέση που πρέπει. Η διαδικασία αυτή ονομάζεται *εγγραφή* (write) στη μνήμη. Μετά από μία εγγραφή, το περιεχόμενο μίας θέσης της μνήμης διατηρείται το ίδιο έως ότου γίνει νέα εγγραφή στην ίδια θέση. Τότε το παλιό της περιεχόμενο χάνεται και αντικαθίσταται με το νέο.

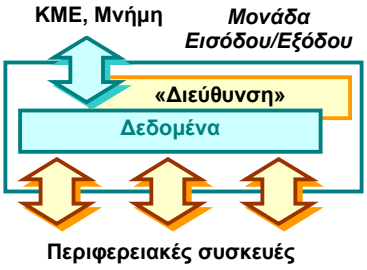
Για να διαβαστεί το περιεχόμενο μίας θέσης της μνήμης, πρέπει πρώτα να γραφεί στον MAR η διεύθυνσή της. Μόλις ενεργοποιηθεί η μονάδα της μνήμης, τα περιεχόμενα της θέσης αυτής μεταφέρονται στον MDR, για να χρησιμοποιηθούν μετά από κάποια άλλη μονάδα του υπολογιστή. Αυτή η διαδικασία ονομάζεται *ανάγνωση* (read). Η ανάγνωση δεν επηρεάζει το περιεχόμενο της θέσης μνήμης που διαβάζεται.

Η μονάδα μνήμης περιέχει επιπλέον κυκλώματα για τον έλεγχο της λειτουργίας της και την επικοινωνία με τις υπόλοιπες μονάδες του υπολογιστή.

Μονάδα Εισόδου - Εξόδου

Η μονάδα εισόδου - εξόδου δέχεται πληροφορίες από το εξωτερικό περιβάλλον του υπολογιστή, μέσω των περιφερειακών μονάδων και δίνει πάλι μέσω των μονάδων τα αποτελέσματα σε αυτό.

Όπως και η μονάδα μνήμης, η μονάδα εισόδου / εξόδου διαθέτει τουλάχιστον δύο καταχωρητές, έναν για τα δεδομένα που μεταδίδονται από και προς τις περιφερειακές συσκευές, και έναν για τη «διεύθυνση» της περιφερειακής συσκευής με την οποία ανταλλάσσονται τα δεδομένα. Τα δεδομένα αποθηκεύονται προσωρινά σε μικρές «μνήμες» της μονάδας που ονομάζονται *απομονωτές* (buffers).

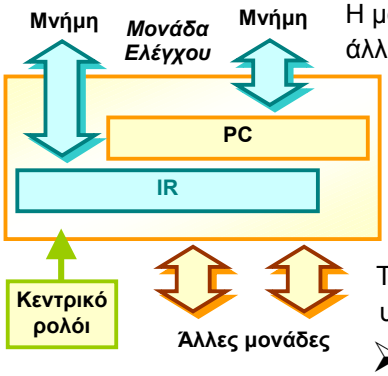


Το μέγεθος του καταχωρητή των δεδομένων της μονάδας Ε/Ε είναι συνήθως ίδιο με το μήκος λέξης του υπολογιστή, ενώ το μέγεθος του καταχωρητή των «διευθύνσεων» εξαρτάται από το πλήθος των περιφερειακών που μπορεί να χειριστεί η μονάδα.

Οι περιφερειακές μονάδες έχουν γενικά μικρή ταχύτητα, πολύ μικρότερη από την ΚΜΕ. Για να γίνει συγχρονισμός της διαφοράς ταχύτητας μεταξύ ΚΜΕ και περιφερειακών μονάδων, υπάρχουν μέσα στη μονάδα εισόδου-εξόδου ειδικά κυκλώματα για κάθε περιφερειακό, τα οποία ελέγχουν εάν έχει τελειώσει μια μεταφορά δεδομένων από τη μονάδα προς το περιφερειακό (ή αντίστροφα).

Επίσης, η μονάδα εισόδου-εξόδου περιέχει διάφορα κυκλώματα για τον έλεγχο της λειτουργίας της και την επικοινωνία της με τις υπόλοιπες μονάδες του υπολογιστή.

Μονάδα ελέγχου



Η μονάδα ελέγχου αποτελείται από δύο κύριους καταχωρητές, καθώς και από άλλα απαραίτητα κυκλώματα για τον έλεγχο και συντονισμό της λειτουργίας του υπολογιστή.

- *Καταχωρητής εντολών* (instruction register - IR). Ο καταχωρητής αυτός που δέχεται μία προς μία τις εντολές του προγράμματος από τη μνήμη, για να αναγνωρισθούν, να αναλυθούν σε επί μέρους εργασίες και τέλος να εκτελεσθούν.

Το μήκος του καταχωρητή εντολών IR είναι ίσο με το μήκος λέξεως του υπολογιστή.

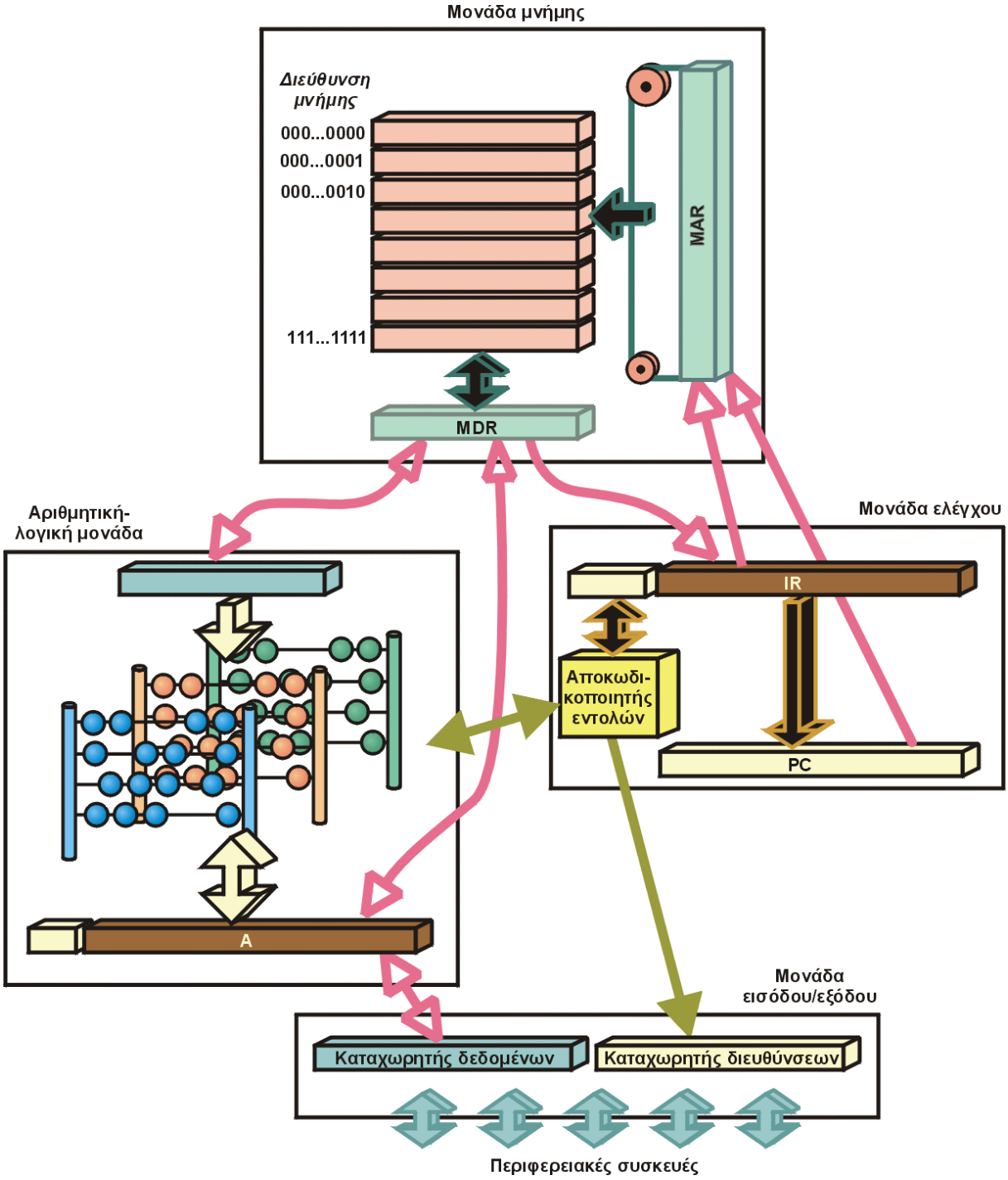
- *Μετρητής προγράμματος* ή *μετρητής εντολών* (program counter – PC, instruction counter). Το περιεχόμενό του δίνει κάθε φορά τη διεύθυνση κάποιας θέσης της μνήμης στην οποία υπάρχει η επόμενη εντολή προγράμματος που θα εκτελεστεί.

Η εντολή αυτή θα μεταφερθεί πρώτα από τη μνήμη στον καταχωρητή IR, και μετά

ο μετρητής προγράμματος θα αυξηθεί αυτομάτως κατά 1 για να αναφέρεται στην επόμενη εντολή.
Αν 2^k είναι το πλήθος των θέσεων της μνήμης που μπορεί να έχει ο υπολογιστής, το μήκος του μετρητή προγράμματος PC είναι k.

Η μονάδα ελέγχου δέχεται σε τακτικά χρονικά διαστήματα, της τάξης των λίγων ns, ένα σήμα από το κεντρικό ρολόι του υπολογιστή (ένα κρύσταλλο). Στο «ρυθμό» αυτού του σήματος γίνονται όλες οι λειτουργίες στον υπολογιστή.

Η γενική οργάνωση του υπολογιστή φαίνεται και στο σχήμα που ακολουθεί.



Ανακεφαλαίωση

Οι βασικές μονάδες ενός υπολογιστή είναι τέσσερις:

- *Η αριθμητική-λογική μονάδα που εκτελεί όλες τις αριθμητικές και λογικές πράξεις που περιέχονται σε ένα πρόγραμμα.*
- *Η μονάδα μνήμης, η οποία αποθηκεύει τα προγράμματα και τα δεδομένα τους.*
- *Η μονάδα εισόδου/εξόδου, η οποία συντονίζει την επικοινωνία του υπολογιστή με τις περιφερειακές συσκευές, και, μέσω αυτών, με το χρήστη.*
- *Η μονάδα ελέγχου η οποία αποκωδικοποιεί τα προγράμματα που εκτελούνται από τον υπολογιστή και αποφασίζει ποιες λειτουργίες πρέπει να εκτελέσουν οι υπόλοιπες μονάδες.*

Η αριθμητική-λογική μονάδα και η μονάδα ελέγχου αποτελούν την Κεντρική Μονάδα Επεξεργασίας του Υπολογιστή.

Γλωσσάριο όρων

Ανάγνωση	*Read*
Απομονωτής	*Buffer*
Αριθμητική-Λογική Μονάδα	*Arithmetic Logical Unit - ALU*
Δεδομένα Εισόδου	*Input Data*
Δεδομένα Εξόδου	*Output Data*
Διεύθυνση	*Address*
Εγγραφή	*Write*
Επεξεργαστής	*Processor*
Θέση Μνήμης	*Memory Position*
Καταχωρητής	*Register*
Καταχωρητής Δεδομένων Της Μνήμης	*Memory Data Register - MDR*
Καταχωρητής Διευθύνσεων Της Μνήμης	*Memory Address Register - MAR*
Καταχωρητής Εντολών	*Instruction Register - IR*
Κεντρική Μονάδα Επεξεργασίας - ΚΜΕ	*Central Processing Unit - CPU*
Λέξη	*Word*
Λογικό Αθροισμα	*OR*
Λογικό Γινόμενο	*AND*
Μετρητής Προγράμματος ή Μετρητής Εντολών	*Program Counter - PC ή Instruction Counter*
Μονάδα Εισόδου-Εξόδου	*Input/Output ή I/O Unit*
Μονάδα Ελέγχου	*Control Unit*
Μονάδα Μνήμης	*Memory Unit*
Ολίσθηση	*Shift*
Πρόγραμμα	*Program*
Συσσωρευτής	*Accumulator*

Ερωτήσεις

? Ποια είναι τα κύρια μέρη ενός ηλεκτρονικού υπολογιστή;

? Ποια είναι η οργάνωση και η λειτουργία της αριθμητικής-λογικής μονάδας;

? Ποια είναι η οργάνωση και η λειτουργία της μονάδας μνήμης;

? Ποια είναι η οργάνωση και η λειτουργία της μονάδας εισόδου/εξόδου;

? Ποια είναι η οργάνωση και η λειτουργία της μονάδας ελέγχου;

? Τι ονομάζουμε μήκος λέξης;

? Ποια είναι η λειτουργία των καταχωρητών MAR και MDR;

? Ποια είναι η λειτουργία του καταχωρητή εντολών και του μετρητή προγράμματος;

? Περίγραψε τη διαδικασία της εγγραφής και της ανάγνωσης στη μνήμη.

Μάθημα 3.2 Γλώσσα Μηχανής

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τις εντολές που μπορεί να εκτελέσει η ΚΜΕ ενός εκπαιδευτικού υπολογιστή και τον τρόπο που μπορούμε να γράψουμε προγράμματα με αυτές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς την αρχή των δύο φάσεων*
- *Να απαριθμείς τις εντολές ενός εκπαιδευτικού υπολογιστή και να εξηγείς τη λειτουργία τους*
- *Να γράφεις απλά προγράμματα για τον υπολογιστή αυτόν και να εξηγείς τα αποτελέσματα της εκτέλεσής τους*

Άβακας: ο αρχαιότερος αριθμητικός υπολογιστής

Στο μάθημα αυτό και στα επόμενα, θα περιγράψουμε τον τρόπο λειτουργίας ενός υπολογιστή με τη βοήθεια ενός εκπαιδευτικού υπολογιστή που ονομάζεται Άβακας. Ο Άβακας είναι σκόπιμα μικρός και απλός για εκπαιδευτικούς λόγους, ωστόσο οι αρχές λειτουργίας του είναι οι ίδιες με αυτές των πραγματικών υπολογιστών.

Το μήκος λέξης του Άβακα είναι 16 bits. Στην Αριθμητική-Λογική του μονάδα περιέχει το συσσωρευτή Α. Οι αρνητικοί αριθμοί στον Άβακα κωδικοποιούνται με την παράσταση συμπληρώματος του 2.

Η αρχή των δύο φάσεων

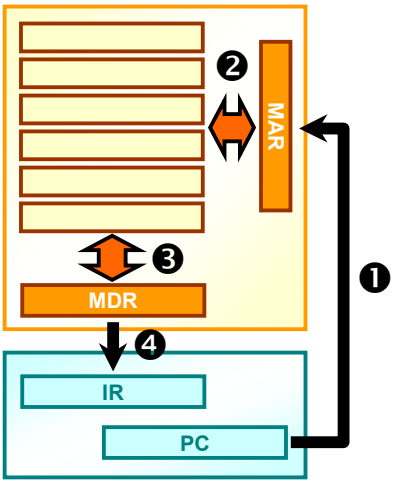
Όπως έχουμε ήδη μάθει, στη μνήμη αποθηκεύονται πληροφορίες που μπορούν να ερμηνευθούν σαν δεδομένα ή σαν εντολές. Για να εκτελεστεί ένα πρόγραμμα από τον υπολογιστή, πρέπει οι εντολές του προγράμματος να είναι αποθηκευμένες σε κάποιες από τις θέσεις της μνήμης. Μάλιστα, συνήθως οι εντολές αποθηκεύονται σε διαδοχικές θέσεις της μνήμης σύμφωνα με την σειρά που πρέπει να εκτελεστούν.

Υπάρχει λοιπόν μια ομάδα διευθύνσεων μνήμης οι οποίες αντιστοιχούν σε θέσεις μνήμης που περιέχουν εντολές. Θα αποκαλούμε «διεύθυνση της εντολής» τη διεύθυνση της θέσης μνήμης που περιέχει την εντολή.

Για να εκτελέσει κάθε εντολή του προγράμματος, ο υπολογιστής περνά από δύο *φάσεις* (phases) ή *κύκλους* (cycles):

- Τη *φάση* (ή *κύκλο*) *ανάκλησης* (fetch phase ή cycle). Στη φάση αυτή μεταφέρεται από τη μνήμη στην ΚΜΕ η εντολή που θα εκτελεστεί.
- Τη *φάση* (ή *κύκλο*) *εκτέλεσης* (execution phase ή cycle), όπου γίνεται η εκτέλεση των λειτουργιών που υποδηλώνει η συγκεκριμένη εντολή.

Φάση ανάκλησης



Στη φάση ανάκλησης, η διεύθυνση εντολής που περιέχεται στο μετρητή προγράμματος PC μεταφέρεται στον καταχωρητή διευθύνσεων μνήμης MAR της μονάδας μνήμης (βήμα ❶). Στη συνέχεια ενεργοποιείται η λειτουργία ανάγνωσης της μονάδας μνήμης (βήμα ❷) και το περιεχόμενο της θέσης που υποδεικνύει ο MAR, δηλαδή η εντολή, μεταφέρεται στον καταχωρητή δεδομένων μνήμης MDR (βήμα ❸). Από εκεί, η εντολή μεταφέρεται στον καταχωρητή εντολών IR της μονάδας ελέγχου (βήμα ❹).

Όταν τελειώσει η μεταφορά της εντολής στον IR, ο μετρητής προγράμματος PC αυξάνεται αυτομάτως κατά 1, και υποδεικνύει την επόμενη εντολή στη μνήμη. Έτσι είναι έτοιμος για την ανάκληση της επόμενης εντολής.

Ο καταχωρητής εντολών IR περιέχει πλέον την τρέχουσα εντολή, η οποία είναι έτοιμη να εκτελεστεί από την ΚΜΕ.

Φάση εκτέλεσης

Στη συνέχεια ακολουθεί η φάση εκτέλεσης. Σε αυτή η εντολή που βρίσκεται στον καταχωρητή εντολών IR αποκωδικοποιείται από ένα ειδικό κύκλωμα που ονομάζεται *αποκωδικοποιητής* (decoder) και αναλύεται σε επί μέρους στοιχειώδεις λειτουργίες.

Η μονάδα ελέγχου στέλνει τότε στις υπόλοιπες μονάδες τα κατάλληλα σήματα για την εκτέλεση των επί μέρους λειτουργιών με την κατάλληλη σειρά. Η ολοκλήρωση των επί μέρους λειτουργιών σημαίνει και εκτέλεση της ίδιας της εντολής.

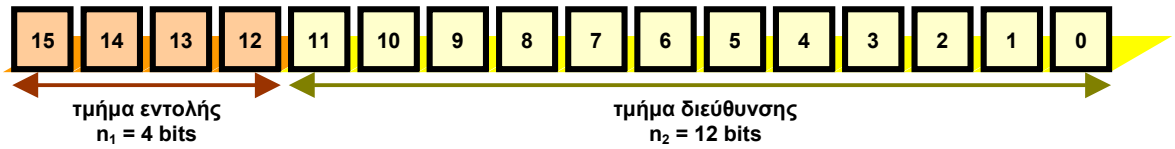
Τη φάση εκτέλεσης ακολουθεί ένα νέο ζεύγος φάσης ανάκλησης - φάσης εκτέλεσης για την επόμενη εντολή.

Τα ζεύγη των δύο φάσεων θα επαναλαμβάνονται για όλες τις εντολές του προγράμματος με τη σειρά, ώσπου να ανακληθεί και εκτελεστεί μια συγκεκριμένη εντολή του υπολογιστή, η οποία υποδεικνύει τη διακοπή της λειτουργίας του (εντολή STOP).

Στην πράξη, στους πιο πολλούς υπολογιστές υπάρχουν και άλλες δευτερεύουσες φάσεις μέσα στον κύκλο κάθε εντολής.

Μορφή της εντολής

Κάθε εντολή παριστάνεται με μία ή περισσότερες λέξεις του υπολογιστή, ενώ μερικοί υπολογιστές αποθηκεύουν πολλές εντολές σε μία λέξη. Στον Άβακα κάθε εντολή έχει μέγεθος μίας λέξης, δηλαδή 16 bits.



Η εντολή χωρίζεται σε δύο τμήματα, όπως βλέπουμε και στο σχήμα. Το πρώτο ονομάζεται *κώδικας εντολής* (instruction code) και έχει μήκος n_1 bits και το δεύτερο *τμήμα διευθύνσεως* (address part) και έχει μήκος n_2 bits· το άθροισμα $n_1 + n_2$ δίνει το μήκος της εντολής. Ο κώδικας εντολής ορίζει τη λειτουργία που πρέπει να εκτελεστεί από την ΚΜΕ, ενώ το τμήμα διευθύνσεως περιέχει τα δεδομένα της.

Στον Άβακα, που το μήκος της εντολής είναι 16 bits, τα n_1 = 4 πρώτα bits περιέχουν τον κώδικα της εντολής και τα υπόλοιπα n_2 = 12 bits αποτελούν το τμήμα διευθύνσεως.

Ο σχεδιαστής κάθε υπολογιστή αποφασίζει και καθορίζει τις εντολές που θα μπορεί να εκτελέσει ο υπολογιστής, δηλαδή το *ρεπερτόριο εντολών* του (instruction set). Στη συνέχεια αντιστοιχίζει κάθε μία από τις εντολές με ένα δυαδικό αριθμό των n_1 bits που θα τη συμβολίζει.

Αφού ο κώδικας εντολής έχει μήκος n_1 bits και κάθε κώδικας πρέπει βέβαια να παριστάνει μόνο μία εντολή, κάθε υπολογιστής μπορεί να διαθέτει έως 2^{n_1} διαφορετικές εντολές.

Στον Άβακα όπου n_1=4 bits, ο μέγιστος αριθμός των διαφορετικών εντολών είναι 2^4=16.

Ο πίνακας που περιέχει τις εντολές ενός υπολογιστή και τον αντίστοιχο δυαδικό κώδικα για κάθε εντολή είναι ο *πίνακας εντολών* του υπολογιστή αυτού.

Οι εντολές χωρίζονται σε ομάδες, ανάλογα με την εργασία την οποία επιτελούν. Υπάρχουν λοιπόν εντολές εκτέλεσης αριθμητικών πράξεων, εντολές μεταφοράς πληροφοριών, εντολές άλματος, εντολές ολίσθησης, εντολές εισόδου-εξόδου κλπ.

Το τμήμα διευθύνσεως της εντολής, που έχει μήκος n_2, είναι ένας δυαδικός αριθμός, που συνήθως παριστάνει μια διεύθυνση μνήμης.

Η εντολή 0011000000001101 του Άβακα, έχει κώδικα εντολής 0011 και τμήμα διευθύνσεως $000000001101_{(2)}$. Η τιμή αυτή, αν ερμηνευθεί σαν διεύθυνση μνήμης, δηλώνει τη θέση μνήμης με διεύθυνση $1101_{(2)} = 13_{(10)}$.

Οι έννοιες «διεύθυνση της εντολής» και «τμήμα διευθύνσεως της εντολής» είναι διαφορετικές και δεν πρέπει να τις συγχέουμε.

Διεύθυνση της εντολής είναι η διεύθυνση μιας θέσεως της μνήμης στην οποία είναι *αποθηκευμένη* η εντολή.

Το **τμήμα διευθύνσεως της εντολής** είναι ένα μέρος της ίδιας της εντολής, το οποίο συνήθως υποδεικνύει τη διεύθυνση κάποιας *άλλης* θέσεως της μνήμης, το περιεχόμενο της οποίας θα χρησιμοποιηθεί κατά την εκτέλεση της εντολής.

Εντολές αναφοράς στη μνήμη

Θα ξεκινήσουμε με το ρεπερτόριο εντολών του Άβακα από μία ομάδα εντολών που ονομάζονται *εντολές αναφοράς στη μνήμη* (memory reference instructions). Στις εντολές αυτές, το τμήμα διεύθυνσης ερμηνεύεται πάντα σαν μία διεύθυνση της μνήμης.

Στον πίνακα που ακολουθεί θα συμβολίζουμε:

- την τιμή του τμήματος διεύθυνσης της εντολής με **N**
- το περιεχόμενο του συσσωρευτή A με **(A)**
- το περιεχόμενο της διεύθυνσης που υποδεικνύει το τμήμα διεύθυνσης N της εντολής με **(N)**

- την αποθήκευση μίας τιμής σε μία θέση μνήμης ή σε ένα καταχωρητή με το σύμβολο ←

Κώδικας εντολής	Σημασία της εντολής	
0001	(A) ← (N)	**Το περιεχόμενο της διεύθυνσης N της μνήμης αντιγράφεται στο συσσωρευτή A.**
0010	(N) ← (A)	**Το περιεχόμενο του συσσωρευτή A αντιγράφεται στη διεύθυνση N της μνήμης.**
0011	(A) ← (A)+(N)	**Το περιεχόμενο της διεύθυνσης N της μνήμης προστίθεται στο περιεχόμενο του συσσωρευτή A. Το αποτέλεσμα μένει στο συσσωρευτή και αποτελεί το νέο περιεχόμενό του.**
0100	(A) ← (A)-(N)	**Το περιεχόμενο της διεύθυνσης N της μνήμης αφαιρείται από το περιεχόμενο του συσσωρευτή A. Το αποτέλεσμα μένει στο συσσωρευτή.**
0101	(A) ← (A)×(N)	**Το περιεχόμενο του συσσωρευτή A πολλαπλασιάζεται με το περιεχόμενο της διεύθυνσης N της μνήμης. Το αποτέλεσμα μένει στο συσσωρευτή.**
0110	(A) ← (A):(N)	**Το περιεχόμενο του συσσωρευτή A διαιρείται με το περιεχόμενο της διεύθυνσης N της μνήμης. Το ακέραιο πηλίκο της διαίρεσης μένει στο συσσωρευτή, ενώ το υπόλοιπο χάνεται.**

Η εντολή 0010000000000010 έχει κώδικα εντολής 0001 και τμήμα διεύθυνσης $000000000010_{(2)} = 2_{(10)}$. Έτσι η εντολή σημαίνει «αντίγραψε το περιεχόμενο του συσσωρευτή A στη θέση 2 της μνήμης».

Η εντολή 0100000000000110 έχει κώδικα εντολής 0100 και τμήμα διεύθυνσης $000000000110_{(2)} = 6_{(10)}$. Η εντολή σημαίνει «αφαίρεσε το περιεχόμενο της θέσης 6 της μνήμης από το συσσωρευτή A και αποθήκευσε το αποτέλεσμα πάλι στον A». Το περιεχόμενο της θέσης 6 της μνήμης δεν αλλάζει.

Ας δούμε ένα μικρό πρόγραμμα που χρησιμοποιεί εντολές αναφοράς στη μνήμη.

Διεύθυνση μνήμης	Περιεχόμενο: Κώδικας εντολής	Περιεχόμενο: Τμήμα διεύθυνσης	
000000000000	0001	000000001010	Πρόγραμμα
000000000001	0011	000000001011	
000000000010	0010	000000001100	
000000000011	0000	000000001000	
000000000100	**Το περιεχόμενο αυτών των θέσεων μνήμης δε μας ενδιαφέρει.**		
000000000101			
000000000110			
000000000111			
000000001000			
000000001001			
000000001010	0000000010000000		Δεδομένα προγράμματος
000000001011	0000000010000101		
000000001100	0000000000000000		

Το πρόγραμμα αυτό αποτελείται από τέσσερις εντολές, αποθηκευμένες στις διευθύνσεις της μνήμης $0_{(10)}$ - $3_{(10)}$.

Εντολή 0: Ο κώδικας εντολής είναι 0001 και το τμήμα διευθύνσεως έχει την τιμή $1010_{(2)} = 10_{(10)}$. Η εντολή σημαίνει «φόρτωσε το συσσωρευτή A με το περιεχόμενο της θέσης μνήμης $10_{(10)}$».

Εντολή 1: Ο κώδικας εντολής είναι 0011 και το τμήμα διευθύνσεως έχει την τιμή $1011_{(2)} = 11_{(10)}$. Η εντολή σημαίνει «πρόσθεσε στο συσσωρευτή A με το περιεχόμενο της θέσης μνήμης $11_{(10)}$».

Εντολή 2: Ο κώδικας εντολής είναι 0010 και το τμήμα διευθύνσεως έχει την τιμή $1100_{(2)} = 12_{(10)}$. Η εντολή σημαίνει «αποθήκευσε το περιεχόμενο του συσσωρευτή A στη θέση μνήμης $12_{(10)}$».

Εντολή 3: Ο κώδικας εντολής είναι 0000, που όπως θα δούμε παρακάτω σημαίνει «τερμάτισε τη λειτουργία του υπολογιστή», είναι δηλαδή η εντολή STOP.

Θα παρακολουθήσουμε βήμα-βήμα την εκτέλεση του προγράμματος αυτού. Αρχικά, οι τρεις καταχωρητές της ΚΜΕ που χρησιμοποιούμε, δηλαδή ο PC, ο IR και ο A έχουν την τιμή 0.

Εντολή	Λειτουργία της εντολής	Καταχωρητές της ΚΜΕ
0 ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000001 (IR) = 0001000000001010 (A) = 0000000000000000
0 εκτέλεση	Ο κώδικας εντολής ($0001_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (N), δηλαδή (A) ← ($1010_{(2)}$). Ο συσσωρευτής A φορτώνεται με το περιεχόμενο της θέσης μνήμης $1010_{(2)}$ με τα εξής βήματα: 1) Η διεύθυνση μνήμης N = 000000001010 μεταφέρεται στον καταχωρητή MAR της μονάδας μνήμης. 2) Ενεργοποιείται η μονάδα μνήμης για ανάγνωση και τα περιεχόμενα της θέσης μνήμης N γράφονται στον καταχωρητή δεδομένων MDR. 3) Από τον MDR τα δεδομένα που έχουν την τιμή $0000000010000000_{(2)}$ αντιγράφονται στο συσσωρευτή A.	(PC) = 000000000001 (IR) = 0001000000001010 (A) = 0000000010000000
1 ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000010 (IR) = 0011000000001011 (A) = 0000000010000000
1 εκτέλεση	Ο κώδικας εντολής ($0011_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (A)+(N), δηλαδή (A) ← (A)+($1011_{(2)}$). Ο υπολογισμός γίνεται με τα εξής στάδια: 1) Η διεύθυνση μνήμης N = 000000001011 μεταφέρεται στον καταχωρητή MAR της μονάδας μνήμης. 2) Ενεργοποιείται η μονάδα μνήμης για ανάγνωση και τα περιεχόμενα της θέσης μνήμης N γράφονται στον καταχωρητή δεδομένων MDR. 3) Το περιεχόμενο του MDR που έχει την τιμή $0000000010000101_{(2)}$ προστίθεται μέσα στην αριθμητική-λογική μονάδα του Άβακα με τα περιεχόμενα του A. Το αποτέλεσμα έχει την τιμή $0000000100000101_{(2)}$ και είναι αποθηκευμένο μέσα στην αριθμητική-λογική μονάδα. 4) Το αποτέλεσμα της πρόσθεσης καταχωρείται στο συσσωρευτή A.	(PC) = 000000000010 (IR) = 0011000000001011 (A) = 0000000100000101
2 ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000011 (IR) = 0010000000001100 (A) = 0000000100000101
2 εκτέλεση	Ο κώδικας εντολής ($0010_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (N) ← (A), δηλαδή ($1100_{(2)}$) ← (A). Το περιεχόμενο του συσσωρευτή καταχωρείται στη θέση μνήμης $1100_{(2)}$ με τα εξής βήματα: 1) Η διεύθυνση μνήμης N = 000000001100 μεταφέρεται στον καταχωρητή MAR της μονάδας μνήμης. 2) Το περιεχόμενο του συσσωρευτή A που είναι $0000000100000101_{(2)}$ αντιγράφεται στον MDR. 3) Ενεργοποιείται η μονάδα μνήμης για εγγραφή και τα περιεχόμενα του καταχωρητή δεδομένων MDR γράφονται στη θέση μνήμης N.	(PC) = 000000000011 (IR) = 0010000000001100 (A) = 0000000100000101
3 ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000100 (IR) = 0000000000001000 (A) = 0000000100000101
3 εκτέλεση	Ο κώδικας εντολής ($0000_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή STOP. Η λειτουργία του υπολογιστή σταματά. Το τμήμα διευθύνσεως της εντολής αγνοείται.	

Μετά την ολοκλήρωση του προγράμματος το περιεχόμενο της θέσης μνήμης $1100_{(2)}$ = $12_{(10)}$ έχει αλλάξει και έχει πάρει την τιμή $0000000100000101_{(2)}$. Οι υπόλοιπες θέσεις μνήμης διατηρούν τα περιεχόμενά τους.

Στο πρόγραμμα αυτό οι θέσεις μνήμης με διευθύνσεις από 000000000000 έως και 000000000011 περιέχουν εντολές, ενώ οι θέσεις μνήμης με διευθύνσεις από 000000001010 έως και 000000001100 περιέχουν δεδομένα. Ο υπολογιστής όμως δεν κάνει διάκριση μεταξύ δεδομένων και εντολών. Αν κάποια στιγμή ο μετρητής

προγράμματος PC είχε την τιμή 000000001011, το περιεχόμενο της αντίστοιχης θέσης μνήμης θα είχε εκτελεστεί σαν να ήταν εντολή, παρότι στην πραγματικότητα αποτελεί δεδομένα.

> Είναι ευθύνη του προγραμματιστή να γράψει το πρόγραμμα κατά τέτοιο τρόπο, ώστε ο υπολογιστής να εκτελεί τις εντολές και όχι κατά λάθος δεδομένα σαν εντολές.

Η εντολή STOP

Η εντολή STOP έχει κωδικό 0000 και η εκτέλεσή της προκαλεί τον τερματισμό της λειτουργίας του Άβακα.

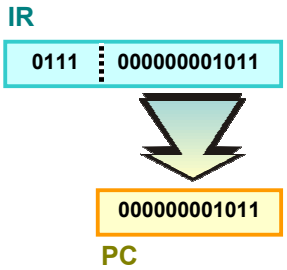
Εντολές άλματος

> Ας φανταστούμε ένα πρόγραμμα αρκετά μεγάλο, ώστε να καταλαμβάνει όλες τις θέσεις της μνήμης του Άβακα, που είναι 2^{12} = 4096. Αν κάθε εντολή του Άβακα απαιτεί κατά μέσο όρο 2ms[1] για να εκτελεστεί, τότε όλο το πρόγραμμα θα χρειαζόταν 4096 · 2ms = 8,192s για να εκτελεστεί. Ξέρουμε όμως πάρα πολλά προγράμματα που απαιτούν πολύ περισσότερο χρόνο για να εκτελεστούν σε πολύ πιο γρήγορους υπολογιστές. Τι συμβαίνει;

Στο ρεπερτόριο εντολών όλων των υπολογιστών υπάρχουν εντολές οι οποίες αλλάζουν τη σειρά εκτέλεσης των εντολών. Όταν εκτελεστεί μια τέτοια εντολή, αλλάζει η τιμή του μετρητή προγράμματος και η εκτέλεση του προγράμματος συνεχίζεται σε κάποιο άλλο σημείο του, σε μεγαλύτερη ή μικρότερη διεύθυνση μνήμης. Αυτές είναι οι *εντολές άλματος* (jump instructions) και είναι απαραίτητες για τη δημιουργία προγραμμάτων, όπου κάποιες εντολές εκτελούνται κατ' επανάληψη.

Ο Άβακας διαθέτει δύο εντολές άλματος:

Α) Την *εντολή άλματος χωρίς συνθήκη* (unconditional jump instruction) με κώδικα εντολής 0111.



Κατά την εκτέλεση της εντολής αυτής το περιεχόμενο του τμήματος διευθύνσεως του καταχωρητή εντολών IR γράφεται στο μετρητή προγράμματος PC.

Στην επόμενη φάση ανακλήσεως θα διαβαστεί από τη μνήμη η επόμενη εντολή, της οποίας η διεύθυνση περιέχεται στο μετρητή προγράμματος. Το περιεχόμενο του μετρητή προγράμματος όμως τώρα έχει μεταβληθεί από την εντολή άλματος και είναι το ίδιο με το τμήμα διευθύνσεως της εντολής άλματος χωρίς συνθήκη.

Διεύθυνση μνήμης	Περιεχόμενο: Κώδικας εντολής	Περιεχόμενο: Τμήμα διεύθυνσης
000000000000	0111	000000000010
000000000001	0000	000000001011
000000000010	0111	000000000000
000000000011	0000	000000001000

Ας δούμε τι θα γίνει, όταν εκτελεστεί το πρόγραμμα του σχήματος. Αρχικά ο μετρητής προγράμματος έχει την τιμή 0. Έτσι κατά τη φάση ανάκλησης

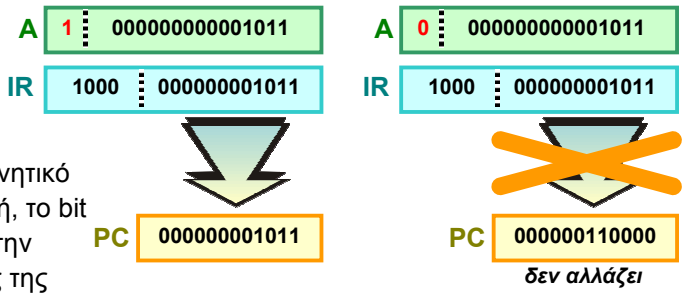
[1] Τα 2ms στην πραγματικότητα είναι πολύ μεγάλος χρόνος για μια εντολή. Οι σύγχρονοι επεξεργαστές εκτελούν χιλιάδες εντολές σ' αυτό το χρονικό διάστημα.

διαβάζεται από τη μνήμη η εντολή που βρίσκεται στη θέση μνήμης 000000000000, και ο μετρητής προγράμματος αυξάνεται κατά 1, παίρνοντας την τιμή 000000000001. Η εντολή που θα εκτελεστεί είναι η 0111 000000000010. Πρόκειται για μια εντολή άλματος χωρίς συνθήκη και η εκτέλεσή της έχει σαν αποτέλεσμα την εγγραφή στον PC του τμήματος διευθύνσεως του IR, το οποίο είναι το 000000000010. Η τιμή του PC λοιπόν αλλάζει και γίνεται 000000000010.

Η εκτέλεση της εντολής ολοκληρώθηκε και ήρθε η ώρα για την επόμενη φάση ανάκλησης. Τώρα το περιεχόμενο του PC «δείχνει» στην τρίτη εντολή του προγράμματος, και όχι στη δεύτερη όπως θα γινόταν αν δεν είχε εκτελεστεί η εντολή άλματος.

Β) Την *εντολή άλματος υπό συνθήκη αρνητικού περιεχομένου του συσσωρευτή* (conditional jump instruction) με κώδικα εντολής 1000.

Όταν εκτελείται η εντολή άλματος υπό συνθήκη αρνητικού περιεχομένου του συσσωρευτή, το αποτέλεσμα εξαρτάται από την τιμή του συσσωρευτή Α.



- Αν το περιεχόμενο του συσσωρευτή είναι αρνητικό (δηλαδή το αριστερότερο bit του συσσωρευτή, το bit του προσήμου, είναι 1), εκτελείται το άλμα στην εντολή που υποδεκνύει το τμήμα διεύθυνσης της εντολής, γράφοντας στο μετρητή προγράμματος το περιεχόμενο του τμήματος διεύθυνσης.
- Αν το περιεχόμενο του συσσωρευτή είναι 0 ή θετικός αριθμός, δηλαδή το αριστερότερό του bit έχει την τιμή 0, τότε η εντολή δεν προκαλεί την εκτέλεση καμιάς λειτουργίας.

Ας δούμε την εκτέλεση ενός προγράμματος, που υπολογίζει την απόλυτη τιμή του αριθμού που βρίσκεται στη θέση $20_{(10)} = 10100_{(2)}$ της μνήμης και την αποθηκεύει στη θέση $50_{(10)} = 110010_{(2)}$ της μνήμης.

Στο σχήμα βλέπουμε ότι ο αριθμός που είναι αποθηκευμένος στη θέση 20 είναι θετικός, αφού το πιο σημαντικό του ψηφίο έχει την τιμή 0. Έτσι, όταν εκτελεστεί το πρόγραμμα, η θέση 50 της μνήμης πρέπει να περιέχει τον αριθμό αυτό.

Διεύθυνση μνήμης	Περιεχόμενο: Κώδικας εντολής	Περιεχόμενο: Τμήμα διεύθυνσης
000000000000	0001	000000010100
000000000001	1000	000000000100
000000000010	0010	000000110010
000000000011	0000	000000001000
000000000100	0001	000000000111
000000000101	0100	000000010100
000000000110	0111	000000000010
000000000111	0000	000000000000
000000010100	0000000010000000	
000000110010	0000000000000000	

Οι εντολές του προγράμματος είναι οι εξής:

Εντολή 0: Ο κώδικας εντολής είναι 0001 και το τμήμα διευθύνσεως έχει την τιμή $10100_{(2)} = 20_{(10)}$. Η εντολή σημαίνει «φόρτωσε το συσσωρευτή Α με το περιεχόμενο της θέσης μνήμης 20».

Εντολή 1: Ο κώδικας εντολής είναι 1000 και το τμήμα διευθύνσεως έχει την τιμή $100_{(2)} = 4_{(10)}$. Αυτή είναι μια εντολή άλματος υπό συνθήκη, και σημαίνει «αν ο Α είναι αρνητικός, πήγαινε στην εντολή 4, αλλιώς μην κάνεις τίποτα».

Εντολή 2: Ο κώδικας εντολής είναι 0010 και το τμήμα διευθύνσεως έχει την τιμή $110010_{(2)} = 50_{(10)}$, πρόκειται δηλαδή για την εντολή «αποθήκευσε το περιεχόμενο του συσσωρευτή Α στη θέση μνήμης 50».

Εντολή 3: Η εντολή αυτή έχει κώδικα 0000, είναι δηλαδή μία εντολή STOP.

Εντολή 4: Έχει κώδικα 0001 και τμήμα διευθύνσεως $111_{(2)} = 7_{(10)}$. Η εντολή αυτή θα φέρει στον Α το περιεχόμενο της θέσης 7 της μνήμης, η οποία όπως βλέπουμε έχει την τιμή 0. Έτσι πρακτικά η εντολή αυτή θέτει τον Α στην τιμή 0.

Εντολή 5: Η εντολή αυτή με κωδικό 0100 αφαιρεί από το συσσωρευτή (που έχει την τιμή 0) το περιεχόμενο της θέσης μνήμης 20, υπολογίζοντας έτσι την αντίθετη τιμή του.

Εντολή 6: Ο κώδικας εντολής είναι 0111 0010 και το τμήμα διευθύνσεως έχει την τιμή $10_{(2)} = 2_{(10)}$, πρόκειται δηλαδή για άλμα χωρίς συνθήκη στην εντολή 2.

Εντολή		Λειτουργία της εντολής	Καταχωρητές της ΚΜΕ
0	ανά-κληση	**Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.**	(PC) = 000000000001 (IR) = 0001000000010100 (A) = 0000000000000000
	εκτέλεση	**Ο κώδικας εντολής ($0001_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (N), δηλαδή (A) ← ($10100_{(2)}$). Ο συσσωρευτής Α φορτώνεται με το περιεχόμενο της θέσης μνήμης $10100_{(2)}$.**	(PC) = 000000000001 (IR) = 0001000000010100 (A) = 0000000010000000
1	ανά-κληση	**Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.**	(PC) = 000000000010 (IR) = 1000000000000100 (A) = 0000000010000000
	εκτέλεση	**Ο κώδικας εντολής ($1000_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή άλματος υπό συνθήκη. Το αριστερότερο bit του συσσωρευτή είναι 0, έτσι δε γίνεται καμία άλλη λειτουργία.**	(PC) = 000000000010 (IR) = 1000000000000100 (A) = 0000000010000000
2	ανά-κληση	**Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.**	(PC) = 000000000011 (IR) = 0010000000110010 (A) = 0000000010000000
	εκτέλεση	**Ο κώδικας εντολής ($0010_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (N) ← (A), δηλαδή ($110010_{(2)}$) ← (A). Το περιεχόμενο του συσσωρευτή καταχωρείται στη θέση μνήμης $110010_{(2)}$.**	(PC) = 000000000011 (IR) = 0010000000110010 (A) = 0000000010000000
3	ανά-κληση	**Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.**	(PC) = 000000000100 (IR) = 0000000000001000 (A) = 0000000010000000
	εκτέλεση	**Ο κώδικας εντολής ($0000_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή STOP. Η λειτουργία του υπολογιστή σταματά.**	

Μετά την εκτέλεση της τελευταίας εντολής η θέση μνήμης 50 περιέχει την τιμή 0000000010000000, η οποία είναι η απόλυτη τιμή του αριθμού που περιέχεται στη θέση 20 της μνήμης.

Ας δούμε τι γίνεται αν στη θέση μνήμης 20 περιέχεται ένας αρνητικός αριθμός, π.χ. ο $-14_{(10)} = 1111111111110010_{(2)}$.

Εντολή		Λειτουργία της εντολής	Καταχωρητές της ΚΜΕ
0	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000001 (IR) = 0001000000010100 (A) = 0000000000000000
	εκτέλεση	Ο κώδικας εντολής ($0001_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (N), δηλαδή (A) ← ($10100_{(2)}$). Ο συσσωρευτής A φορτώνεται με το περιεχόμενο της θέσης μνήμης $10100_{(2)}$.	(PC) = 000000000001 (IR) = 0001000000010100 (A) = 1111111111110010
1	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000010 (IR) = 1000000000000100 (A) = 1111111111110010
	εκτέλεση	Ο κώδικας εντολής ($1000_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή άλματος υπό συνθήκη. Το αριστερότερο bit του συσσωρευτή είναι 1, έτσι εκτελείται η εντολή άλματος. Το τμήμα διεύθυνσης του IR μεταφέρεται στον PC, που τώρα δείχνει στην εντολή 4 και όχι στη 2 που έδειχνε μετά τη φάση ανάκλησης.	(PC) = 000000000100 (IR) = 1000000000000100 (A) = 1111111111110010
2	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000101 (IR) = 0001000000000111 (A) = 1111111111110010
	εκτέλεση	Ο κώδικας εντολής ($0001_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (N), δηλαδή ($A_{)}$) ← ($111_{(2)}$). Το περιεχόμενο της θέσης μνήμης $111_{(2)}$ μεταφέρεται στον A.	(PC) = 000000000101 (IR) = 0001000000000111 (A) = 0000000000000000
3	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000110 (IR) = 0100000000010100 (A) = 0000000000000000
	εκτέλεση	Ο κώδικας εντολής ($0100_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (A) ← (A) - (N), δηλαδή (A) ← (A) -($10100_{(2)}$). Το περιεχόμενο της θέσης μνήμης $10100_{(2)}$ αφαιρείται από τον A που έχει την τιμή 0, δηλαδή υπολογίζεται ο αντίθετός της. Το αποτέλεσμα καταχωρείται στον A.	(PC) = 000000000110 (IR) = 0100000000010100 (A) = 0000000000001110
4	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000111 (IR) = 0111000000000010 (A) = 0000000000001110
	εκτέλεση	Ο κώδικας εντολής ($0111_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι πρόκειται για εντολή άλματος χωρίς συνθήκη. Το τμήμα διεύθυνσης του IR αντιγράφεται στον PC.	(PC) = 000000000010 (IR) = 0111000000000010 (A) = 0000000000001110
5	ανά-κληση	Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.	(PC) = 000000000011 (IR) = 0010000000110010 (A) = 0000000000001110
	εκτέλεση	Ο κώδικας εντολής ($0010_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή (N) ← (A), δηλαδή ($110010_{(2)}$) ← (A). Το περιεχόμενο του συσσωρευτή καταχωρείται στη θέση μνήμης $110010_{(2)}$.	(PC) = 000000000011 (IR) = 0010000000110010 (A) = 0000000000001110

6 ανά-κληση εκτέλεση	**Το περιεχόμενο της θέσης μνήμης που δείχνει ο PC μεταφέρεται στον IR. Ο PC αυξάνεται κατά 1.** **Ο κώδικας εντολής ($0000_{(2)}$) του IR αποκωδικοποιείται και βρίσκεται ότι είναι η εντολή STOP. Η λειτουργία του υπολογιστή σταματά.**	(PC) = 000000000100 (IR) = 0000000000001000 (A) = 0000000000001110

Αυτή τη φορά η θέση μνήμης 50 περιέχει την τιμή $0000000000001110_{(2)} = 14_{(2)}$, η οποία είναι και πάλι η απόλυτη τιμή του αριθμού που περιέχεται στη θέση 20 της μνήμης.

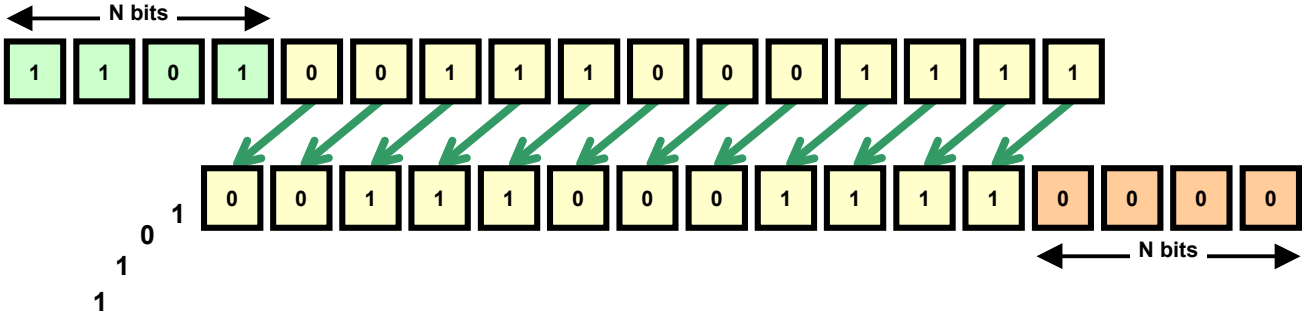
Εντολές ολίσθησης

Ο Άβακας διαθέτει δύο *εντολές ολίσθησης* (shift instructions).

Η πρώτη από αυτές ονομάζεται *εντολή ολίσθησης προς τα αριστερά* (shift left instruction) και έχει κώδικα εντολής 1011.

Η εντολή προκαλεί ολίσθηση του περιεχομένου του συσσωρευτή A κατά N θέσεις προς τα αριστερά, όπου N είναι ο αριθμός που περιέχει το τμήμα διευθύνσεως της εντολής. Από τα δεξιά ο αριθμός συμπληρώνεται με μηδενικά bits.

> Αν ο A περιέχει την τιμή 1101001110001111 και εκτελεστεί η εντολή 1011 000000000100 (ολίσθηση προς τα αριστερά κατά 4 θέσεις), το νέο του περιεχόμενο θα είναι το 0011100011110000.



Για να μηδενίσουμε λοιπόν τον A μπορούμε να τον ολισθήσουμε προς τα αριστερά κατά 16 θέσεις, οπότε όλα του τα ψηφία θα χαθούν και θα αντικατασταθούν από μηδενικά.

Η δεύτερη εντολή ολίσθησης είναι η *εντολή ολίσθησης προς τα δεξιά* (shift right instruction) με κώδικα εντολής 1100. Η εντολή προκαλεί ολίσθηση του περιεχομένου του συσσωρευτή A κατά N θέσεις προς τα δεξιά, όπου N είναι ο αριθμός που περιέχει το τμήμα διευθύνσεως της εντολής. Από τα αριστερά ο αριθμός συμπληρώνεται με μηδενικά bits.

> Αν ο A περιέχει την τιμή **1101001110001**111 και εκτελεστεί η εντολή 1100 000000000100 (ολίσθηση προς τα δεξιά κατά 4 θέσεις), το νέο του περιεχόμενο θα είναι το 0000**110100111000**.

> Στις εντολές ολίσθησης το τμήμα διευθύνσεως δεν περιέχει μια διεύθυνση αλλά τον αριθμό των ολισθήσεων που θα πραγματοποιηθούν.

Εντολές εισόδου-εξόδου

Η επικοινωνία του ανθρώπου με τον υπολογιστή πραγματοποιείται με τις περιφερειακές μονάδες, μέσω της μονάδας εισόδου-εξόδου (ή *μονάδας Ε/Ε*). Υπάρχουν πολλών ειδών περιφερειακές μονάδες στους υπολογιστές: πληκτρολόγιο, οθόνη, ποντίκι, εκτυπωτής, κλπ.

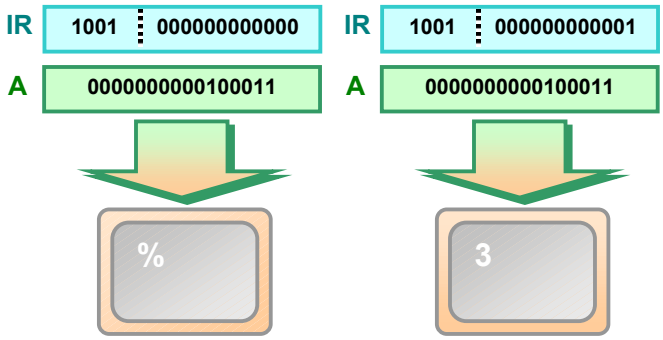
Ο Άβακας διαθέτει ένα πληκτρολόγιο και μια οθόνη ως περιφερειακές μονάδες που είναι συνδεδεμένα με την μονάδα Ε/Ε. Για να επικοινωνεί με τις περιφερειακές μονάδες αυτές, και μέσω αυτών με το χρήστη, ο Άβακας περιλαμβάνει στο ρεπερτόριο εντολών του δύο *εντολές Ε/Ε* (input /output instructions ή I/O instructions).

Η πρώτη είναι η *εντολή εξόδου* (output instruction) με κώδικα εντολής 1001.

- Αν το δεξιότερο bit της εντολής, δηλαδή το λιγότερο σημαντικό ψηφίο της, έχει την τιμή 0, τότε τα 7 δεξιότερα bits του συσσωρευτή A (bits 0-6) στέλνονται στην οθόνη όπου εμφανίζεται ο αντίστοιχος χαρακτήρας της κωδικοποίησης ASCII.
- Αν το δεξιότερο bit της εντολής έχει την τιμή 1, τότε τα 3 δεξιότερα bits του συσσωρευτή A (bits 0-2) στέλνονται στην οθόνη όπου εμφανίζεται το αντίστοιχο οκταδικό ψηφίο.

Αν ο συσσωρευτής έχει την τιμή 0000000000**0100011**, η εντολή 1001000000000000 προκαλεί την εμφάνιση στην οθόνη του χαρακτήρα που αντιστοιχεί στον κωδικό 0100011, δηλαδή του **%**.

Η εντολή 1001000000000001 θα έχει σαν αποτέλεσμα την εμφάνιση του οκταδικού αριθμού που αντιστοιχεί στα bits **011**, δηλαδή του **3**.

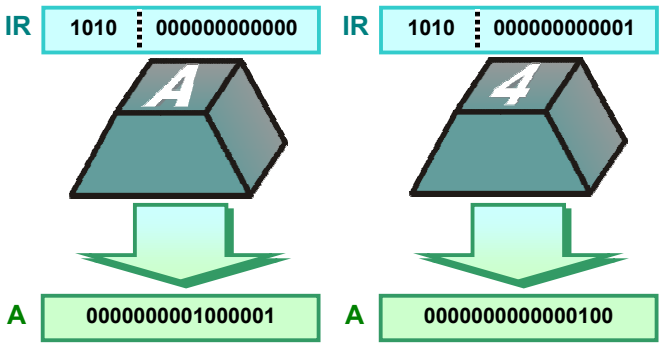


Η δεύτερη εντολή I/O είναι η *εντολή εισόδου* (input instruction) με κώδικα εντολής 1010.

Με την εντολή αυτή διαβάζεται από το πληκτρολόγιο μια πληροφορία και καταχωρείται στο συσσωρευτή. Ο υπολογιστής αναμένει το χρήστη να πατήσει ένα πλήκτρο, οπότε αποθηκεύεται στον καταχωρητή δεδομένων της μονάδας Ε/Ε ένας χαρακτήρας ASCII. Η πληροφορία που θα αποθηκευθεί στον A εξαρτάται από το δεξιότερο bit της εντολής:

- Αν το δεξιότερο bit της εντολής, δηλαδή το λιγότερο σημαντικό ψηφίο της, έχει την τιμή 0, τα 7 δεξιότερα bits του καταχωρητή δεδομένων της μονάδας Ε/Ε αποθηκεύονται στα 7 δεξιότερα bits του A.
- Αν το δεξιότερο bit της εντολής, δηλαδή το λιγότερο σημαντικό ψηφίο της, έχει την τιμή 1, τα 3 δεξιότερα bits του καταχωρητή δεδομένων της μονάδας Ε/Ε αποθηκεύονται στα 3 δεξιότερα bits του A και παριστάνουν έναν οκταδικό αριθμό. Αυτό γίνεται ακόμα και αν το πλήκτρο που πάτησε ο χρήστης δεν είναι ένα οκταδικό ψηφίο.

Και στις εντολές Ε/Ε το τμήμα διευθύνσεως της εντολής δεν δείχνει κάποια διεύθυνση· χρησιμοποιείται μόνο το δεξιότερο bit του για τον προσδιορισμό του τύπου των δεδομένων που μεταφέρονται, αριθμού ή χαρακτήρα.



Όταν εκτελεστεί η εντολή 1010000000000000, ο υπολογιστής περιμένει το χρήστη να πατήσει κάποιο πλήκτρο. Αν ο χρήστης πατήσει το πλήκτρο A, που έχει ASCII κώδικα $65_{(10)}$, στα 7 τελευταία bits του A θα καταχωρισθεί η τιμή $1000001_{(2)}$.

Αν εκτελεστεί η εντολή 1010000000000001, και ο χρήστης πατήσει το πλήκτρο 4, στα 3 τελευταία bits του A θα καταχωρισθεί η τιμή $100_{(2)}$ = 4.

Ανακεφαλαίωση

Η εκτέλεση μίας εντολής μηχανής από ένα υπολογιστή περιλαμβάνει τη φάση ανάκλήσης της εντολής στη μνήμη και τη φάση εκτέλεσης της εντολής.

Ο εκπαιδευτικός υπολογιστής Άβακας περιέχει ένα καταχωρητή στην αριθμητική-λογική μονάδα του, το συσσωρευτή A. Διαθέτει 13 εντολές μηχανής, που αποτελούνται από εντολές αναφοράς στη μνήμη (περιλαμβάνουν και τις τέσσερις αριθμητικές πράξεις), εντολές άλματος χωρίς και υπό συνθήκη, εντολές ολίσθησης και εντολές εισόδου-εξόδου.

Γλωσσάριο όρων

Αποκωδικοποιητής	*Decoder*
Εντολή Άλματος	*Jump Instruction*
Εντολή Άλματος Υπό Συνθήκη	*Conditional Jump Instruction*
Εντολή Άλματος Χωρίς Συνθήκη	*Unconditional Jump Instruction*
Εντολή Αναφοράς Στη Μνήμη	*Memory Reference Instruction*
Εντολή Ε/Ε	*Input /Output Instruction ή I/O Instruction*
Εντολή Εισόδου	*Input Instruction*
Εντολή Εξόδου	*Output Instruction*
Εντολή Ολίσθησης	*Shift Instruction*
Εντολή Ολίσθησης Προς Τα Αριστερά	*Shift Left Instruction*
Εντολή Ολίσθησης Προς Τα Δεξιά	*Shift Right Instruction*
Κύκλος	*Cycle*
Κώδικας Εντολής	*Instruction Code*
Τμήμα Διευθύνσεως	*Address Part*
Φάση	*Phase*
Φάση ή Κύκλος Ανάκλησης	*Fetch Phase ή Cycle*
Φάση ή Κύκλος Εκτέλεσης	*Execution Phase ή Cycle*

Ερωτήσεις

? Ποιες είναι οι φάσεις εκτέλεσης μίας εντολής και τα τέσσερα βήματα της φάσης ανάκλησης;

? Τι εννοούμε με τον όρο «διεύθυνση εντολής» και με τον όρο «τμήμα διεύθυνσης εντολής»;

? Ποια είναι η λειτουργία των εντολών άλματος;

? Ποια είναι η λειτουργία των εντολών ολίσθησης;

Μάθημα 3.3 Συμβολική Γλώσσα

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει τον προγραμματισμό ενός υπολογιστή σε συμβολική γλώσσα.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τις διαφορές της συμβολικής γλώσσας από τη γλώσσα μηχανής.*
- *Να περιγράφεις το ρόλο των ψευδοεντολών σε ένα πρόγραμμα.*
- *Να γράφεις απλά προγράμματα στη συμβολική γλώσσα του Άβακα.*

Μέχρι τώρα είδαμε παραδείγματα προγραμματισμού του Άβακα όπου τόσο οι εντολές όσο και οι διευθύνσεις μνήμης ήταν γραμμένες σε αριθμητική μορφή και μάλιστα στο δυαδικό σύστημα. Τα προγράμματα αυτά είναι γραμμένα σε *γλώσσα μηχανής* (machine language).

Είναι φανερό ότι η σύνταξη έστω και πολύ απλών προγραμμάτων στη γλώσσα μηχανής είναι εργασία κουραστική και απαιτεί χρόνο. Τα πράγματα θα ήταν πολύ πιο απλά αν γράφαμε τις εντολές μας με τρόπο «συμβολικό», που να μας θυμίζει αμέσως το νόημά τους.

> Η εντολή 0001 φορτώνει (**L**oa**D**s) το συσσωρευτή (**A**ccumulator) με μία τιμή. Ένα καλό μνημονικό όνομα για την εντολή αυτή είναι το **LDA**.

Μπορούμε επίσης να δίνουμε συμβολικά ονόματα στις διευθύνσεις της μνήμης, ώστε να δουλεύουμε με σύμβολα αντί με αριθμούς.

> Αν συμβολίσουμε τη διεύθυνση $000000100101_{(2)}$ με το όνομα «IN_NUM», τότε η εντολή 0001 000000100101 θα μπορούσε να γραφτεί συμβολικά σαν «**LDA IN_NUM**».

Με τον τρόπο αυτό μπορούμε να γράφουμε τις εντολές του προγράμματος μας σε μια άλλη γλώσσα, που ονομάζουμε *συμβολική γλώσσα* (assembly language).

Ένα πρόγραμμα γραμμένο σε συμβολική γλώσσα, για να μπορεί να εκτελεστεί από τον υπολογιστή, πρέπει να μεταφραστεί πρώτα σε γλώσσα μηχανής. Τη μετάφραση αυτή των προγραμμάτων από συμβολική γλώσσα σε γλώσσα μηχανής την πραγματοποιεί ο ίδιος ο υπολογιστής, με ένα ειδικό πρόγραμμα που είναι γραμμένο σε γλώσσα μηχανής και ονομάζεται *συμβολομεταφραστής* (assembler).

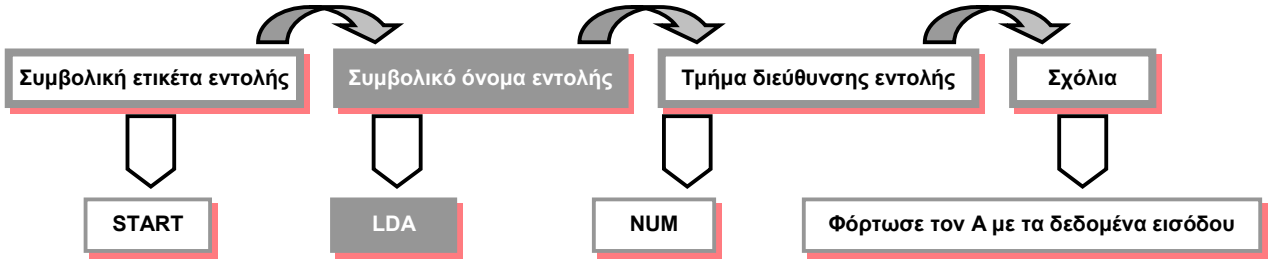
Ο συμβολομεταφραστής, εκτός από το να μετατρέπει το πρόγραμμα από τη συμβολική γλώσσα σε γλώσσα μηχανής, αναγνωρίζει τα λάθη («συντακτικά» ή «ορθογραφικά»), που μπορεί να υπάρχουν στις εντολές του προγράμματός μας.

Για κάθε εντολή από το ρεπερτόριο εντολών του υπολογιστή υπάρχει και η αντίστοιχη εντολή σε συμβολική γλώσσα. Επιπλέον όμως η συμβολική γλώσσα διαθέτει μερικές ακόμα εντολές, τις *ψευδοεντολές* (pseudo instructions). Οι ψευδοεντολές δεν εκτελούνται από τον υπολογιστή, αλλά χρησιμοποιούνται από το συμβολομεταφραστή κατά το μεταφραστικό στάδιο και διευκολύνουν τον προγραμματιστή όταν γράφει το πρόγραμμα.

Ας δούμε ποιες είναι οι συμβολικές μορφές των εντολών του Άβακα:

Συμβολικό όνομα	Κώδικας	Λειτουργία
HLT = **HaLT**	0000	STOP
LDA = **LoaD A**	0001	Φόρτωση του A (A) ← (N)
STA = **STore to A**	0010	Αποθήκευση του A (N) ← (A)
ADA = **ADd to A**	0011	Πρόσθεση στον A (A) ← (A) + (N)
SBA = **SuBtract from A**	0100	Αφαίρεση από τον A (A) ← (A) - (N)
MLA = **MuLtiply A**	0101	Πολλαπλασιασμός με τον A (A) ← (A) * (N)
DVA = **DiVide A**	0110	Διαίρεση του A (A) ← (A) / (N)
JMP = **JuMP**	0111	Άλμα χωρίς συνθήκη (PC) ← N
JAN = **Jump if A is Negative**	1000	Άλμα υπό συνθήκη (PC) ← N αν (A) < 0
OUT = **OUTput**	1001	Έξοδος δεδομένων στην οθόνη
INP = **INPut**	1010	Είσοδος δεδομένων από το πληκτρολόγιο
SAL = **Shift A Left**	1011	Ολίσθηση του A αριστερά
SAR = **Shift A Right**	1100	Ολίσθηση του A δεξιά

Κάθε εντολή σε ένα πρόγραμμα συμβολικής γλώσσας έχει την εξής μορφή:



Συμβολική ετικέτα εντολής: Το τμήμα αυτό της εντολής είναι προαιρετικό και αποτελεί το «όνομα» που δίνουμε στην εντολή για να μπορούμε να αναφερθούμε σε αυτή σε μία άλλη εντολή άλματος.

Συμβολικό όνομα της εντολής: Περιέχει μία από τις συμβολικές εντολές που είδαμε παραπάνω. Αυτό είναι το μόνο τμήμα της εντολής που υπάρχει υποχρεωτικά.

Τμήμα διεύθυνσης της εντολής: Στο τμήμα αυτό γράφουμε τα δεδομένα που αντιστοιχούν στο τμήμα διεύθυνσης της εντολής γλώσσας μηχανής.

Αν πρόκειται για μία εντολή αναφοράς στη μνήμη, αυτό το πεδίο πρέπει να περιέχει μία διεύθυνση στη μνήμη. Η διεύθυνση μπορεί να γραφτεί σαν αριθμός, σαν ετικέτα ή σαν μία αριθμητική παράσταση που συνδυάζει αριθμούς και ετικέτες.

> Η εντολή «STA 10» αποθηκεύει το περιεχόμενο του A στη θέση μνήμης με διεύθυνση 10. Αν αυτή η θέση μνήμης χαρακτηρίζεται από την ετικέτα SUM, τότε μπορούμε να γράψουμε την εντολή σαν «STA SUM» η οποία αποθηκεύει το περιεχόμενο του A στη θέση μνήμης που συμβολίζεται με την ετικέτα SUM. Παρακάτω θα δούμε πώς αντιστοιχίζουμε στο πρόγραμμά μας θέσεις μνήμης για δεδομένα με ετικέτες.
>
> Αν στη συνέχεια θέλουμε να μεταφέρουμε στον A τα περιεχόμενα της επόμενης θέσης μνήμης από τη SUM, θα γράψουμε «LDA SUM+1».

Αν πρόκειται για εντολή άλματος, το τμήμα διεύθυνσης θα πρέπει να αναφέρεται σε μια άλλη εντολή του προγράμματος. Έτσι μπορεί να περιέχει έναν αριθμό, μία ετικέτα εντολής, ή μία αριθμητική παράσταση με αριθμούς και ετικέτες εντολής.

> Η εντολή «JMP START» προκαλεί άλμα στην εντολή με ετικέτα START, ενώ η εντολή «JMP START+1» προκαλεί άλμα στην επόμενη από την εντολή με ετικέτα START.

Η ειδική ετικέτα εντολής «*» συμβολίζει την ίδια την εντολή.

> Ας δούμε το πρόγραμμα:
>
> ```
> MLA   10
> LDA   SUM
> JMP   *-2
> ```
>
> Όταν φθάσουμε στην τελευταία εντολή, θα εκτελεστεί ένα άλμα στην εντολή που βρίσκεται 2 θέσεις πιο πριν απ' αυτή, δηλαδή στην πρώτη εντολή «MLA 10».

Σύγκρινε το πρόγραμμα αυτό με το παράδειγμα της σελίδας 56.

Στις εντολές εισόδου-εξόδου, όπως έχουμε δει, το δεξιότερο bit του τμήματος διεύθυνσης προσδιορίζει, αν τα δεδομένα που θα μεταφερθούν από ή προς την περιφερειακή συσκευή θα ερμηνευθούν σαν αριθμός ή σαν χαρακτήρας. Στη συμβολική γλώσσα το τμήμα διεύθυνσης έχει αντίστοιχα το περιεχόμενο 0 ή 1.

> Με την εντολή:
>
> ```
> INP   1
> ```
>
> διαβάζουμε από το πληκτρολόγιο έναν οκταδικό αριθμό που αποθηκεύεται στο συσσωρευτή A.

Στις εντολές ολίσθησης το τμήμα διεύθυνσης περιέχει τον αριθμό των θέσεων κατά τις οποίες θέλουμε να ολισθήσουμε το συσσωρευτή.

> Η εντολή «SAR 12» ολισθαίνει το συσσωρευτή Α κατά 12 θέσεις προς τα δεξιά.

Σχόλια: Στο τελευταίο τμήμα, που είναι προαιρετικό, γράφουμε ένα κείμενο της επιλογής μας, το οποίο αποτελεί επεξήγηση για τη λειτουργία της εντολής. Ο συμβολομεταφραστής αγνοεί το τμήμα αυτό.

> Στην παρακάτω εντολή, το κείμενο «Αποθήκευση του Α» αναγνωρίζεται από το συμβολομεταφραστή σαν σχόλιο, και έτσι αγνοείται.
>
> ```
> STA   17    Αποθήκευση του Α
> ```

Ψευδοεντολές

Οι ψευδοεντολές δε μεταφράζονται από το συμβολομεταφραστή σε εντολές μηχανής, αλλά χρησιμοποιούνται κατά τη μετάφραση του προγράμματος. Έχουν και αυτές ένα συμβολικό όνομα που τοποθετείται στο αντίστοιχο τμήμα της εντολής αλλά δεν αντιστοιχεί με κάποιο κώδικα εντολής. Οι ψευδοεντολές μπορούν και αυτές να έχουν συμπληρωμένα και τα υπόλοιπα τμήματά τους.

NAM

Στην αρχή του προγράμματος γράφουμε μία ψευδοεντολή NAM (από τη λέξη name = όνομα). Στο τμήμα διεύθυνσής της γράφουμε μία λέξη που αποτελεί το όνομα του προγράμματος.

> Το πρόγραμμα που υπολογίζει την απόλυτη τιμή ενός αριθμού θα ξεκινά με την ψευδοεντολή «NAM ABS» (από τη λέξη absolute = απόλυτος).

CON

Για να ορίσουμε θέσεις μνήμης που περιέχουν κάποιους αριθμούς και έχουν ένα συμβολικό όνομα, χρησιμοποιούμε την ψευδοεντολή CON (από τη λέξη constant = σταθερά). Στην ψευδοεντολή αυτή πρέπει να δώσουμε μία ετικέτα, για να μπορούμε να αναφερόμαστε στις θέσεις μνήμης αυτές, και αντί για πεδίο διεύθυνσης γράφουμε μία σειρά από αριθμούς.

Η πραγματική διεύθυνση μνήμης που αντιστοιχεί στην ετικέτα δε μας ενδιαφέρει· την επιλέγει ο συμβολομεταφραστής.

> Με την ψευδοεντολή «DATA CON 0,8,27» ορίζουμε τρεις θέσεις μνήμης, που περιέχουν τους αριθμούς 0, 8 και 27 αντίστοιχα. Στην πρώτη θέση αναφερόμαστε ως «DATA», στη δεύτερη ως «DATA+1» και στην τρίτη ως «DATA+3».
>
> Αν μέσα στο πρόγραμμά μας υπάρχει η εντολή «LDA DATA+1», αυτή θα μεταφέρει στον Α το περιεχόμενο της θέσης μνήμης DATA+1, δηλαδή την τιμή 8.

RES

Όπως συμβολίζουμε θέσεις μνήμης που περιέχουν δεδομένα, μπορούμε να συμβολίσουμε θέσεις μνήμης όπου θα αποθηκευθούν τα αποτελέσματα που υπολογίζει το πρόγραμμα. Αυτό γίνεται με μία ψευδοεντολή RES (από το αγγλικό ρήμα reserve = κρατώ, κάνω κράτηση).

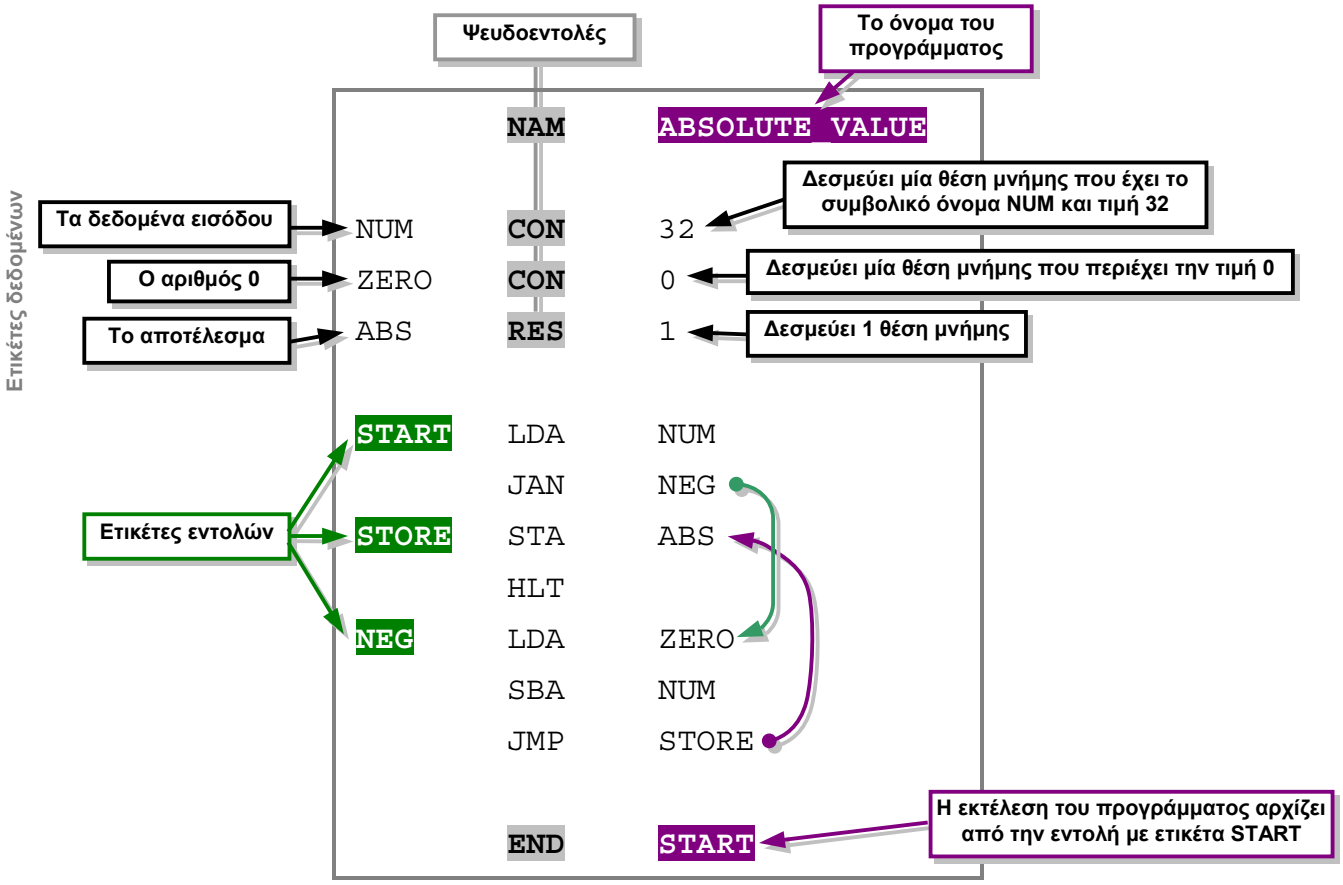
Στην ψευδοεντολή RES δίνουμε μία ετικέτα, για να αναφερόμαστε στις θέσεις μνήμης που κρατεί. Το πεδίο διεύθυνσής της περιέχει έναν αριθμό, το πλήθος των θέσεων μνήμης που θέλουμε να κρατήσουμε.

> Με την ψευδοεντολή «SUM RES 2» κρατάμε δύο θέσεις μνήμης για να αποθηκεύσουμε αποτελέσματα. Για να καταχωρήσουμε το περιεχόμενο του A στην πρώτη από αυτές τις θέσεις, θα γράψουμε την εντολή «STA SUM», ενώ για να το αποθηκεύσουμε στη δεύτερη θέση θα γράψουμε «STA SUM+1».

Το τέλος του προγράμματος συμβολίζεται από μία ψευδοεντολή END.Στο τμήμα διεύθυνσής της γράφουμε την ετικέτα της πρώτης εντολής του προγράμματος. Έτσι μπορούμε στην αρχή του προγράμματος σε συμβολική γλώσσα να τοποθετήσουμε δεδομένα, αλλά η εκτέλεσή του να αρχίσει από το σημείο που θα υποδείξουμε με την ψευδοεντολή END.

> Δεν πρέπει να μπερδέψουμε την ψευδοεντολή END με την εντολή HLT. Η πρώτη απευθύνεται μόνο στο συμβολομεταφραστή και υποδεικνύει την ολοκλήρωση της μετάφρασης. Η δεύτερη απευθύνεται και στον υπολογιστή και προκαλεί παύση της λειτουργίας του.

Ας δούμε ένα πλήρες πρόγραμμα γραμμένο σε συμβολική γλώσσα, που υπολογίζει την απόλυτη τιμή ενός αριθμού. Ο αριθμός βρίσκεται στη θέση μνήμης που έχει οριστεί με την ψευδοεντολή CON και έχει ετικέτα NUM, και το αποτέλεσμα αποθηκεύεται στη θέση μνήμης που έχει οριστεί με την ψευδοεντολή RES και έχει ετικέτα ABS. Στο πρόγραμμα αυτό οι εντολές άλματος χρησιμοποιούνται για τη διακλάδωση της ροής εκτέλεσής του.



Πρόγραμμα με βρόχο

Οι εντολές άλματος έχουν μία πολύ σημαντική χρησιμότητα: βοηθούν να κατασκευάζουμε προγράμματα που εκτελούν επαναληπτικά κάποιες λειτουργίες.

> Η επαναληπτική εκτέλεση μίας ομάδας εντολών σε ένα πρόγραμμα ονομάζεται *βρόχος* (loop).

Αν μάλιστα η επαναληπτική εκτέλεση του βρόχου δε σταματά, τότε λέμε ότι το πρόγραμμα έχει πέσει σε *ατέρμονα βρόχο* (infinite loop).

Μία διαδικασία που μπορεί να υλοποιηθεί με ένα βρόχο, είναι η ανάγνωση από το πληκτρολόγιο ενός αριθμού, bit προς bit. Η διαδικασία ανάγνωσης ενός bit είναι η ίδια, ανεξάρτητα από τη θέση του μέσα στη λέξη. Έτσι μπορούμε να κατασκευάσουμε ένα βρόχο που θα επαναληφθεί 16 φορές και θα διαβάζει ένα bit από το πληκτρολόγιο.

```
INP  1
SAL  15
SAR  15
```

Επειδή από το πληκτρολόγιο μπορούμε να διαβάσουμε ή χαρακτήρες ή οκταδικούς αριθμούς, και όχι μόνο 1 bit, θα επιλέξουμε την ανάγνωση οκταδικών αριθμών, αλλά θα χρησιμοποιούμε μόνο το τελευταίο δυαδικό ψηφίο τους. Για να το κάνουμε αυτό, διαβάζουμε τον οκταδικό αριθμό στο συσσωρευτή A, τον ολισθαίνουμε 15 θέσεις προς τα αριστερά (οπότε όλα τα ψηφία του εκτός από το τελευταίο χάνονται) και μετά τον ολισθαίνουμε 15 θέσεις προς τα δεξιά, οπότε επανέρχεται το τελευταίο bit στη σωστή θέση και συμπληρώνεται από τα αριστερά με μηδενικά.

> Αν ο συσσωρευτής A περιέχει αρχικά τον αριθμό 0010000001010001 και διαβαστεί ο οκταδικός αριθμός $3_{(8)}=011_{(2)}$ από το πληκτρολόγιο, η νέα τιμή του A θα είναι 0010000001010011. Ολισθαίνοντας τον A κατά 15 θέσεις προς τα αριστερά, θα πάρει τη νέα τιμή 1000000000000000. Όταν τον ολισθήσουμε ξανά 15 θέσεις προς τα δεξιά, η τιμή του θα γίνει 0000000000000001, έχει διατηρηθεί δηλαδή μόνο το δεξιότερο bit.

Ο αριθμός που διαβάζουμε θα αποθηκευθεί σε μία θέση μνήμης στην οποία θα δώσουμε το όνομα NUM. Αρχικά η θέση μνήμης NUM θα έχει την τιμή 0. Επίσης θα χρειαστούμε άλλη μία θέση μνήμης, με το όνομα COUNTER, στην οποία θα μετράμε το πλήθος των επαναλήψεων, ώστε να σταματήσουμε μετά από 16 επαναλήψεις, όσες και τα bits μίας λέξης. Και η μεταβλητή COUNTER αρχικά θα έχει την τιμή 0. Σε γενικές γραμμές, η λειτουργία του προγράμματος θα είναι η εξής:

Βήμα 1: Φόρτωσε την τρέχουσα τιμή του αριθμού από τη θέση μνήμης NUM.

Βήμα 2: Ολίσθησε προς τα αριστερά την τρέχουσα τιμή του αριθμού κατά 1, για να κάνεις «χώρο» για το νέο bit.

Βήμα 3: Αποθήκευσε την τρέχουσα τιμή του αριθμού στη θέση μνήμης NUM.

Βήμα 4: Διάβασε ένα bit από το πληκτρολόγιο.

Βήμα 5: Πρόσθεσε την τρέχουσα τιμή του αριθμού NUM στο bit αυτό, για να πάρεις το νέο αριθμό.

Βήμα 6: Αποθήκευσε τη νέα τιμή του αριθμού στη θέση μνήμης NUM.

Βήμα 7: Φόρτωσε την τιμή του μετρητή COUNTER στο συσσωρευτή.

Βήμα 8: Πρόσθεσε 1 στην τιμή του μετρητή.

Βήμα 9: Αποθήκευσε τη νέα τιμή του συσσωρευτή στη θέση μνήμης COUNTER.

Βήμα 10: Αφαίρεσε 16 από την τιμή του συσσωρευτή.

Βήμα 11: Αν η τιμή του A είναι αρνητική, τότε το πρόγραμμα πρέπει να τελειώσει, οπότε γίνεται άλμα στην τελευταία εντολή του, το βήμα 13.

Βήμα 12: Αν φθάσουμε στο βήμα αυτό, το πρόγραμμα δεν έχει τελειώσει ακόμα, έτσι πάμε ξανά στο βήμα 1.

Βήμα 13: STOP.

Ας δούμε και ολόκληρο το πρόγραμμα σε συμβολική γλώσσα:

```
          NAM       READ_NUMBER

NUM       CON       0
COUNTER   CON       0
ONE       CON       1
REPS      CON       16

START     LDA       NUM
          SAL       1
          STA       NUM
          INP       1
          SAL       15
          SAR       15
          ADA       NUM
          STA       NUM
          LDA       COUNTER
          ADA       ONE
          STA       COUNTER
          SBA       REPS
          JAN       START
FINISH    HLT

END       START
```

Η θέση μνήμης όπου αποθηκεύεται ο αριθμός

Η θέση μνήμης όπου αποθηκεύεται ο μετρητής των επαναλήψεων

Ο αριθμός 1, που προσθέτουμε κάθε φορά στο μετρητή

Το πλήθος των επαναλήψεων που θα κάνει το πρόγραμμα

Βήματα 1-3: ολίσθηση του τρέχοντος αριθμού κατά μία θέση αριστερά

Βήμα 4: ανάγνωση ενός bit από το πληκτρολόγιο

Βήματα 5,6: το νέο bit συνδυάζεται με τον αριθμό και αποθηκεύεται στη θέση NUM

Βήματα 7-9: ο μετρητής των επαναλήψεων αυξάνεται κατά 1

Βήμα 10: αφαιρούμε 16 από τη νέα τιμή του μετρητή· αν έχει αρνητική τιμή, αυτό σημαίνει ότι ο μετρητής ήταν μικρότερος από 16, άρα δεν έχουμε τελειώσει ακόμα

Ανακεφαλαίωση

Επειδή ο προγραμματισμός σε γλώσσα μηχανής είναι δύσκολος, γράφουμε τις εντολές του προγράμματος με συμβολικά ονόματα, δηλαδή σε συμβολική γλώσσα. Ένα ειδικό πρόγραμμα, ο συμβολομεταφραστής αναλαμβάνει να μετατρέψει το πρόγραμμα από τη συμβολική γλώσσα σε γλώσσα μηχανής. Ο συμβολομεταφραστής επίσης παρέχει στον προγραμματιστή τη δυνατότητα να ορίζει θέσεις μνήμης με συμβολικά ονόματα, οι οποίες είτε περιέχουν αρχικά κάποιες τιμές, είτε χρησιμοποιούνται για την αποθήκευση αποτελεσμάτων.

Γλωσσάριο όρων

Ατέρμων Βρόχος	*Infinite Loop*
Γλώσσα Μηχανής	*Machine Language*
Συμβολική Γλώσσα	*Assembly Language*
Συμβολομεταφραστής	*Assembler*
Ψευδοεντολή	*Pseudo Instruction*

Ερωτήσεις

- ? Σε τι μας βοηθά ο προγραμματισμός σε συμβολική γλώσσα;
- ? Ποια εντολή παριστάνεται με το συμβολικό όνομα LDA, STA, DVA;
- ? Ποιες εντολές παριστάνονται με τα ονόματα JMP, JAN;
- ? Τι περιέχει το πρώτο πεδίο της εντολής συμβολικής γλώσσας;
- ? Ποια είναι η λειτουργία της ψευδοεντολής NAM;
- ? Σε τι διαφέρει η ψευδοεντολή CON από την ψευδοεντολή RES;
- ? Τι είναι ο βρόχος;

Μάθημα 3.4 Από τη Γλώσσα Μηχανής στα Προγράμματα Εφαρμογής

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει τις τεχνικές και τα εργαλεία που διαθέτουν οι προγραμματιστές για να γράφουν γρήγορα και εύκολα τα προγράμματά τους.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να ορίζεις τις γλώσσες προγραμματισμού*
- *Να περιγράφεις τη λειτουργία του διερμηνέα, του μεταφραστή και του συμβολομεταφραστή*
- *Να εξηγείς πώς λειτουργεί ένας μεταφραστής*

Γλώσσες προγραμματισμού

Ο προγραμματισμός σε γλώσσα μηχανής είναι ένα αρκετά χρονοβόρο έργο, στο οποίο υποβόσκουν κίνδυνοι για πολλά προγραμματιστικά λάθη:

- Είναι πολύ εύκολο να γράψει κανείς ένα λανθασμένο κώδικα εντολής, όταν μάλιστα αυτός αποτελείται από αρκετά bits. Μπορεί να γράψει π.χ. «100101» αντί για «101001» αλλοιώνοντας τελείως το νόημα της εντολής και, βέβαια, το αποτέλεσμα του προγράμματος.
- Επειδή οι εντολές αναφέρονται σε συγκεκριμένες διευθύνσεις μνήμης τόσο για τα δεδομένα τους όσο και για τις εντολές άλματος, ο προγραμματιστής μπορεί να γράψει εύκολα κάποια λανθασμένη διεύθυνση. Τα λάθη αυτά, όπως και τα λάθη στον κώδικα εντολής, είναι δύσκολο και κουραστικό να εντοπιστούν μέσα σε ένα πρόγραμμα.
- Σε όλες τις εντολές άλματος η διεύθυνση προορισμού είναι καθορισμένη με ακρίβεια. Αν όμως το πρόγραμμα μεταβληθεί με την εισαγωγή μίας επιπλέον εντολής ή την αφαίρεση άλλης, όλες οι εντολές άλματος που επηρεάζονται πρέπει να μετατραπούν.

Επίσης όλα τα προγράμματα σε γλώσσα μηχανής περιέχουν ένα πολύ μεγάλο αριθμό εντολών, διότι ακόμα και για τα απλούστερα έργα, όπως μία απλή αριθμητική παράσταση απαιτούνται αρκετές εντολές. Έτσι ο προγραμματιστής σπαταλά πολύ χρόνο γράφοντάς τα και ακόμα περισσότερο ελέγχοντάς τα.

Τα προβλήματα που αναφέρονται λύνονται εν μέρει με τη συμβολική γλώσσα, αν και σε αυτή δεν είναι πολύ δύσκολο να κάνει ένας προγραμματιστής ένα λάθος. Παραμένει όμως το γεγονός ότι τα προγράμματα είναι πολύ ογκώδη, και βέβαια οι προγραμματιστές πρέπει να θυμούνται πάντα τα συμβολικά ονόματα για όλες τις εντολές του υπολογιστή.

Ένα πρόγραμμα σε συμβολική γλώσσα μπορεί να εκτελεστεί μόνο στον τύπο υπολογιστή για τον οποίο γράφτηκε. Για να εκτελεστεί σε οποιοδήποτε άλλο υπολογιστή πρέπει να ξαναγραφεί από την αρχή.

Για να αντιμετωπιστούν όλα αυτά τα προβλήματα, δημιουργήθηκαν οι *γλώσσες προγραμματισμού υψηλού επιπέδου* (high level programming languages). Στις γλώσσες αυτές, που είναι πιο κοντά στην ανθρώπινη γλώσσα και διευκολύνουν τον άνθρωπο-προγραμματιστή, κάθε εντολή αποτελεί μία πολύπλοκη λειτουργία και αντιστοιχεί σε πολλές εντολές γλώσσας μηχανής. Επίσης οι γλώσσες προγραμματισμού δεν εξαρτώνται από κάποια συγκεκριμένη αρχιτεκτονική υπολογιστή.

Μεταφραστές

Οι γλώσσες προγραμματισμού κάνουν το έργο των προγραμματιστών πιο εύκολο, αλλά δεν μπορούν να εκτελεστούν από τον υπολογιστή, που «καταλαβαίνει» μόνο εντολές μηχανής. Το ίδιο ισχύει βέβαια και για τα προγράμματα σε συμβολική γλώσσα. Το χάσμα αυτό γεφυρώνουν οι *μεταφραστές* (translators), ειδικά προγράμματα που μετατρέπουν τα προγράμματα από μία μορφή κατανοητή στον άνθρωπο στη μορφή που μπορεί να εκτελεστεί από τον υπολογιστή.

Οι μεταφραστές διακρίνονται σε τρεις μεγάλες κατηγορίες:

Α. Οι *συμβολομεταφραστές* (assemblers) μετατρέπουν προγράμματα από συμβολική γλώσσα σε γλώσσα μηχανής. Επειδή η αντιστοιχία μεταξύ των συμβολικών εντολών και των εντολών μηχανής είναι ένα προς ένα, το έργο του συμβολομεταφραστή είναι:

   (1) Να αντικαθιστά κάθε συμβολική εντολή με την αντίστοιχη εντολή μηχανής **(2)** να υπολογίζει τις διευθύνσεις κάθε εντολής για να θέτει σωστά τις διευθύνσεις προορισμού στις εντολές άλματος και **(3)** να υπολογίζει τις διευθύνσεις των δεδομένων του προγράμματος και να αντικαθιστά τις ετικέτες που αναφέρονται σε δεδομένα με τις πραγματικές διευθύνσεις.

   Επειδή υπάρχει αυτή η στενή σχέση μεταξύ συμβολικής γλώσσας και γλώσσας μηχανής, ο συμβολομεταφραστής λειτουργεί μόνο στον υπολογιστή για τον οποίο γράφτηκε.

Β. Οι *μεταγλωττιστές* (compilers) διαβάζουν ένα πρόγραμμα σε μία γλώσσα προγραμματισμού και το μετατρέπουν στην αντίστοιχη ακολουθία από εντολές μηχανής. Κάθε εντολή της γλώσσας προγραμματισμού αντικαθίσταται από πολλές εντολές μηχανής.

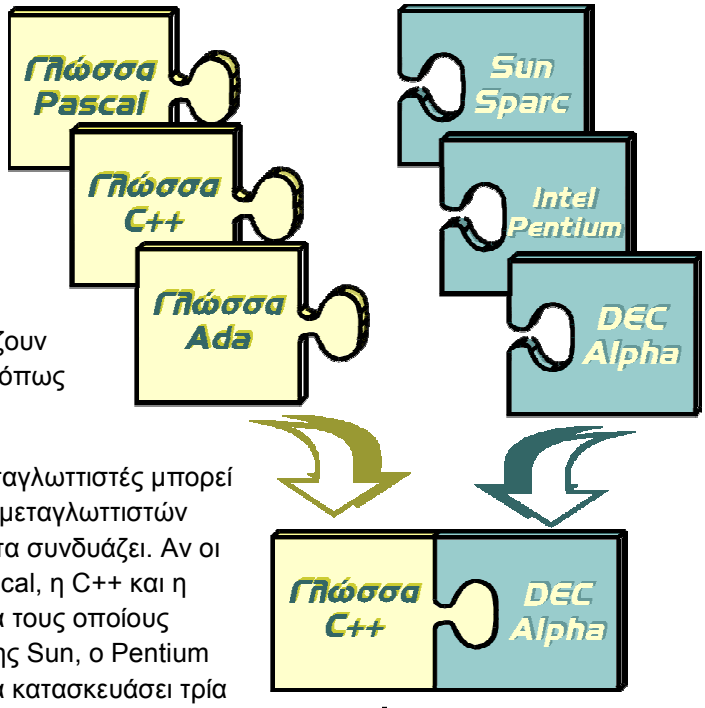
   Ο μεταγλωττιστής είναι πιο πολύπλοκο πρόγραμμα από το συμβολομεταφραστή, γιατί αναγνωρίζει και μετατρέπει πιο πολύπλοκες γλώσσες. Μερικές φορές μάλιστα οι μεταγλωττιστές παράγουν ένα πρόγραμμα σε συμβολική γλώσσα, το οποίο μετά αναλαμβάνει ο συμβολομεταφραστής.

Γ. Οι *διερμηνείς* (interpreters) συγχρόνως μεταφράζουν και εκτελούν ένα πρόγραμμα εντολή προς εντολή. Αυτό σημαίνει ότι κάθε εντολή του προγράμματος μεταφράζεται μόνο όταν έρθει η ώρα να εκτελεστεί, αλλά και ότι μπορεί να μεταφραστεί πολλές φορές, μία για κάθε φορά που εκτελείται.

Δομή του μεταγλωττιστή

Ένας μεταγλωττιστής γενικά αποτελείται από δύο μέρη: το ένα τμήμα είναι αφιερωμένο στην αναγνώριση και ανάλυση του προγράμματος στη γλώσσα υψηλού επιπέδου και το άλλο αναλαμβάνει την παραγωγή της συμβολικής γλώσσας και της

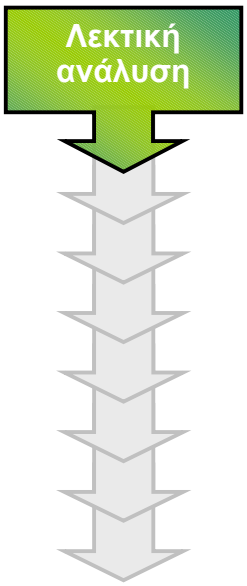
γλώσσας μηχανής. Το τμήμα που κάνει την αναγνώριση της γλώσσας προγραμματισμού δεν εξαρτάται από τον υπολογιστή στον οποίο θα εκτελεστεί αλλά μόνο από τη γλώσσα προγραμματισμού. Αντίστοιχα, το τμήμα που κάνει την παραγωγή της γλώσσας μηχανής δεν εξαρτάται από τη γλώσσα προγραμματισμού αλλά μόνο από το ρεπερτόριο εντολών μηχανής που παράγει. Αυτό δίνει τη δυνατότητα στους κατασκευαστές των μεταγλωττιστών να κατασκευάζουν ανεξάρτητα τα δύο τμήματα, και να τα συνδυάζουν όπως χρειάζεται.



Μία εταιρία λογισμικού που κατασκευάζει μεταγλωττιστές μπορεί να σχεδιάζει και να υλοποιεί τα τμήματα των μεταγλωττιστών ανεξάρτητα το ένα από το άλλο, και μετά να τα συνδυάζει. Αν οι γλώσσες με τις οποίες ασχολείται είναι η Pascal, η C++ και η Ada, ενώ οι αρχιτεκτονικές επεξεργαστών για τους οποίους παράγονται τα προγράμματα είναι ο Sparc της Sun, ο Pentium της Intel και ο Alpha της DEC, τότε μπορεί να κατασκευάσει τρία τμήματα για τις γλώσσες προγραμματισμού και άλλα τρία για τις αρχιτεκτονικές. Συνδυάζοντας τα έξι τμήματα αυτά μεταξύ τους έχει έτοιμους εννέα μεταγλωττιστές.

Το πρώτο τμήμα του μεταγλωττιστή εκτελεί τρεις λειτουργίες: τη λεκτική, τη συντακτική και τη σημασιολογική ανάλυση του προγράμματος. Από τις λειτουργίες αυτές προκύπτει μία αναπαράσταση του προγράμματος η οποία δεν εξαρτάται από τη γλώσσα προγραμματισμού. Την αναπαράσταση αυτή χρησιμοποιεί το δεύτερο μέρος του μεταγλωττιστή, που επιτελεί την παραγωγή κώδικα μηχανής (ή συμβολικής γλώσσας) και τη βελτιστοποίησή του.

Στη *λεκτική ανάλυση* (lexical analysis) του προγράμματος, ο μεταγλωττιστής το χωρίζει στις βασικές του μονάδες: τις λέξεις, τα σύμβολα, τους αριθμούς κλπ. Τα διαστήματα που διαχωρίζουν τις μονάδες του προγράμματος αφαιρούνται, και το αποτέλεσμα είναι μία σειρά από *μονάδες του προγράμματος* (tokens).

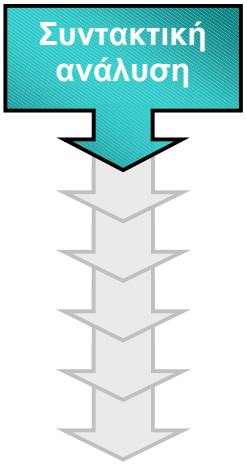


Σε ένα πρόγραμμα Pascal εμφανίζονται οι εντολές

```
y := 5;
x := y+15;
```

Ο λεκτικός αναλυτής θα χωρίσει αυτό το τμήμα του προγράμματος στην ακολουθία:

```
«αναγνωριστικό όνομα y», «τελεστής ανάθεσης»,
«αριθμητική σταθερά 5», «ερωτηματικό»,
«αναγνωριστικό όνομα x», «τελεστής ανάθεσης»,
«αναγνωριστικό όνομα y», «τελεστής συν»,
«αριθμητική σταθερά 15», «ερωτηματικό».
```

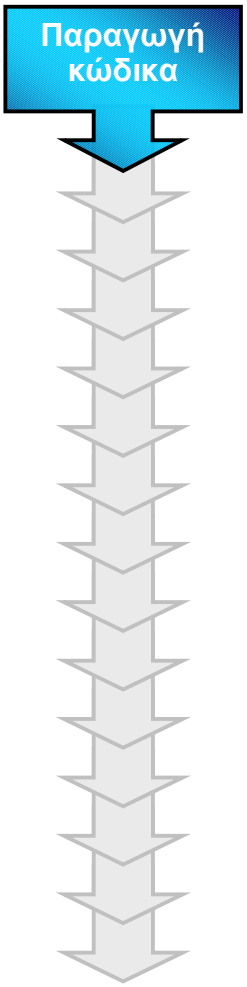


Στη *συντακτική ανάλυση* (syntax analysis) του προγράμματος, ελέγχεται κατά πόσο αυτό ακολουθεί τους συντακτικούς κανόνες της γλώσσας προγραμματισμού, και μετασχηματίζεται σε κάποια ενδιάμεση αναπαράσταση, την *ενδιάμεση γλώσσα* (intermediate language), η οποία δεν εξαρτάται από τη γλώσσα προγραμματισμού.

> Στη γλώσσα Pascal οι εντολές χωρίζονται μεταξύ τους με ερωτηματικό («;»). Ο συντακτικός αναλυτής θα διαπιστώσει το λάθος και θα ενημερώσει τον προγραμματιστή αν έχει κατά λάθος γράψει τις παραπάνω εντολές σαν:
>
> ```
> y := 5
> x := y+15;
> ```
>
> όπου λείπει ένα ερωτηματικό μεταξύ των εντολών.

Παράλληλα με τη συντακτική, γίνεται και η *σημασιολογική ανάλυση* (semantic analysis).

> Αν ο προγραμματιστής γράψει σε ένα πρόγραμμα Pascal:
>
> ```
> y := 5 + "abcde";
> ```
>
> ο σημασιολογικός αναλυτής θα διαπιστώσει την εσφαλμένη πρόσθεση μεταξύ ενός αριθμού και ενός κειμένου, και θα ενημερώσει τον προγραμματιστή για αυτό.



Στη συνέχεια ο μεταγλωττιστής παράγει τις εντολές σε γλώσσα μηχανής ή σε συμβολική γλώσσα οι οποίες αντιστοιχούν στο πρόγραμμα το οποίο μεταφράζει. Αυτό το στάδιο της μεταγλώττισης ονομάζεται *παραγωγή κώδικα* (code generation). Με βάση την ενδιάμεση μορφή του προγράμματος που έχει κατασκευάσει κατά τη συντακτική ανάλυση, ο μεταγλωττιστής αντικαθιστά κάθε εντολή του προγράμματος με τις κατάλληλες εντολές μηχανής. Στο στάδιο αυτό αποφασίζει για τις θέσεις της μνήμης όπου θα αποθηκευθούν τα δεδομένα του προγράμματος.

> Για τις δύο εντολές Pascal που είδαμε, ένας μεταγλωττιστής για τον Άβακα πρέπει να αποφασίσει πού στη μνήμη θα κρατούνται οι δύο σταθερές 5 και 15 και οι δύο μεταβλητές x και y. Αν οι σταθερές τοποθετούνται σε θέσεις μνήμης με συμβολικά ονόματα C5 και C15 αντίστοιχα, και οι δύο μεταβλητές τοποθετούνται σε θέσεις μνήμης με τα συμβολικά ονόματα X και Y, τότε το πρόγραμμα σε συμβολική γλώσσα που θα παραγάγει ο μεταγλωττιστής είναι:
>
> ```
> C5      CON     5
> C15     CON     15
> ...
> X       RES     1
> Y       RES     1
> ...
>
>         LDA     C5
>         STA     Y
>
>         LDA     Y
>         ADA     C15
>         STA     X
> ```
>
> σταθερές
>
> μεταβλητές
>
> y := 5
>
> x :=y+1 5

Οι πληροφορίες για τη θέση των μεταβλητών και των σταθερών κρατούνται από το μεταγλωττιστή στον *πίνακα συμβόλων* (symbol table).

Ο πίνακας συμβόλων για το παράδειγμά μας είναι:

Συμβολική διεύθυνση	Σχετική διεύθυνση	Αρχικό περιεχόμενο
C5	0	0000000000000101
C15	1	0000000000001111
Y	2	0000000000000000
X	3	0000000000000000

Βελτιστοποίηση κώδικα

Το τελευταίο στάδιο στη μεταγλώττιση ενός προγράμματος είναι η *βελτιστοποίηση του κώδικα* (code optimization). Στη φάση αυτή, γίνονται μικρές μετατροπές στον κώδικα μηχανής, με τέτοιο τρόπο ώστε να επιτελεί τις ίδιες λειτουργίες αλλά να είναι πιο γρήγορος ή πιο μικρός σε μέγεθος.

Το πρόγραμμα του παραδείγματός μας, με τη δεύτερη εντολή του αποθηκεύει το περιεχόμενο του συσσωρευτή στη θέση μνήμης με ετικέτα Y και αμέσως μετά αντιγράφει πάλι το περιεχόμενο της θέσης μνήμης Y στο συσσωρευτή. Η δεύτερη αντιγραφή είναι περιττή, αφού η θέση μνήμης Y και ο συσσωρευτής έχουν το ίδιο περιεχόμενο. Κατά τη βελτιστοποίηση του κώδικα, η δεύτερη εντολή θα παραλειφθεί, και το πρόγραμμα θα γίνει τελικά:

```
C5      CON     5
C15     CON     15
X       RES     1
Y       RES     1
START   LDA     C5
        STA     Y
        ADA     C15
        STA     X
        END     START
```

Κώδικας μηχανής

Αν ο μεταγλωττιστής παράγει συμβολική γλώσσα, τότε ο συμβολομεταφραστής θα επεξεργαστεί το πρόγραμμα και θα παραγάγει τις εντολές μηχανής που είναι έτοιμες για εκτέλεση από τον υπολογιστή.

Τελικά, το πρόγραμμα σε γλώσσα μηχανής είναι:

Διεύθυνση	Περιεχόμενο		
000000000000	0001000000000101	LDA	C5
000000000001	0010000000001000	STA	Y
000000000010	0011000000000110	ADA	C15
000000000011	0010000000000111	STA	X
000000000100	0000000000000000	HLT	
000000000101	0000000000000101	C5	
000000000110	0000000000001111	C15	
000000000111	0000000000000000	X	
000000001000	0000000000000000	Y	

Ανακεφαλαίωση

Οι γλώσσες προγραμματισμού υψηλού επιπέδου έχουν δημιουργηθεί για να κάνουν το έργο των προγραμματιστών πιο απλό και γρήγορο. Τα προγράμματα μετατρέπονται από τη γλώσσα προγραμματισμού σε γλώσσα μηχανής από ειδικά προγράμματα: τους μεταφραστές και τους διερμηνείς. Ένας μεταφραστής ακολουθεί συνήθως πέντε στάδια για να μεταφράσει ένα πρόγραμμα: τη λεκτική ανάλυση, τη συντακτική ανάλυση, τη σημασιολογική ανάλυση, την παραγωγή κώδικα και τη βελτιστοποίηση κώδικα.

Γλωσσάριο όρων

Βελτιστοποίηση Κώδικα	*Code Optimization*
Γλώσσα Προγραμματισμού Υψηλού Επιπέδου	*High Level Programming Language*
Διερμηνέας	*Interpreter*
Ενδιάμεση Γλώσσα	*Intermediate Language*
Λεκτική Ανάλυση	*Lexical Analysis*
Μεταγλωττιστής	*Compiler*
Μεταφραστής	*Translator*
Μονάδες Προγράμματος	*Tokens*
Παραγωγή Κώδικα	*Code Generation*
Πίνακας Συμβόλων	*Symbol Table*
Σημασιολογική Ανάλυση	*Semantic Analysis*
Συμβολομεταφραστής	*Assembler*
Συντακτική Ανάλυση	*Syntax Analysis*

Ερωτήσεις

? Ποιες είναι οι δυσκολίες του προγραμματισμού σε γλώσσα μηχανής;

? Τι είναι οι συμβολομεταφραστές και οι μεταφραστές;

? Ποια είναι η διαφορά του μεταγλωττιστή από το διερμηνέα;

? Ποια είναι τα στάδια της μεταγλώττισης;

? Σε τι τμήματα χωρίζει ένα πρόγραμμα η λεκτική ανάλυση;

? Ποιο είναι το έργο της συντακτικής ανάλυσης;

? Ποιο είναι το έργο της σημασιολογικής ανάλυσης;

? Τι γίνεται κατά την παραγωγή κώδικα;

? Ποιος είναι ο σκοπός της βελτιστοποίησης κώδικα;

? Τι περιέχει ο πίνακας συμβόλων;

Μάθημα 3.5 Σύγχρονοι Επεξεργαστές

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τα ιδιαίτερα χαρακτηριστικά των αρχιτεκτονικών που χρησιμοποιούνται στους σύγχρονους επεξεργαστές.

Τι θα μάθεις;

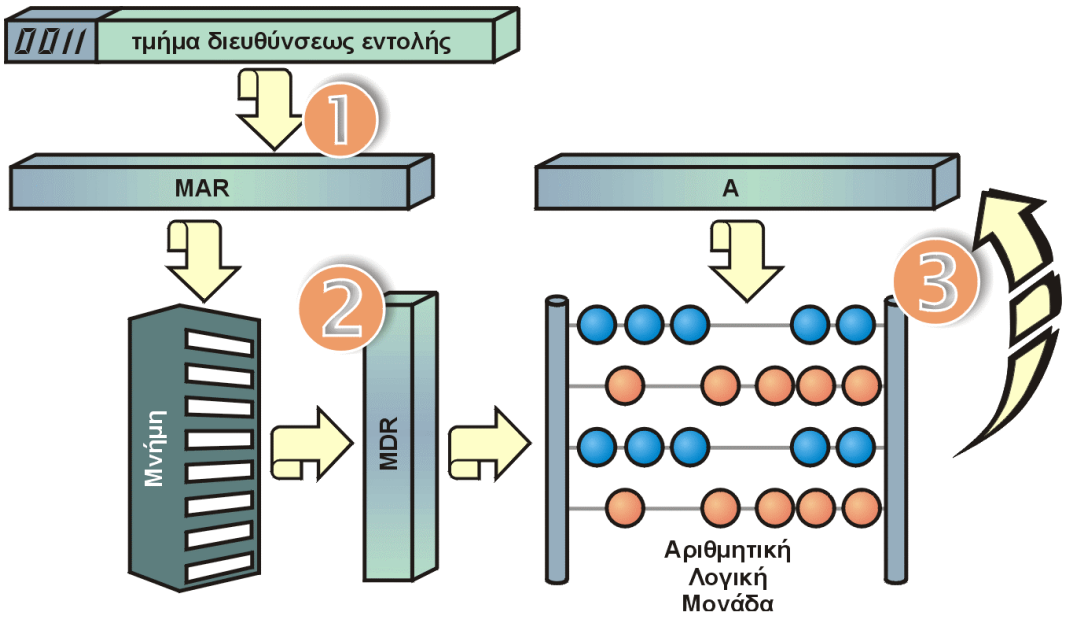
Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τη λειτουργικότητα του μικροπρογραμματισμού*
- *Να περιγράφεις την αρχιτεκτονική RISC και την αρχιτεκτονική αγωγού*
- *Να απαριθμείς τις διάφορες εναλλακτικές αρχιτεκτονικές επεξεργαστών που χρησιμοποιούνται σήμερα*

Μικρολειτουργίες

Γνωρίζουμε ότι κάθε επεξεργαστής μπορεί να εκτελέσει ένα σύνολο από εντολές, το ρεπερτόριο εντολών του. Κάθε εντολή εκτελείται στο εσωτερικό του επεξεργαστή σαν μία σειρά από πολύ απλά και στοιχειώδη βήματα.

Η εντολή του Άβακα με κωδικό 0011 και συμβολικό όνομα ADA, προσθέτει το περιεχόμενο του συσσωρευτή A και το περιεχόμενο της θέσης μνήμης που προσδιορίζεται από το τμήμα διευθύνσεως της εντολής.



Στο σχήμα βλέπουμε τα τρία στάδια εκτέλεσης της εντολής αυτής. Κάθε στάδιο είναι μία στοιχειώδης λειτουργία ή *μικρολειτουργία* (microoperation) του επεξεργαστή.

Το τμήμα διευθύνσεως της εντολής περιέχει τη διεύθυνση μίας θέσης στη μνήμη. Για να διαβαστεί το περιεχόμενο της θέσης αυτής, πρέπει το τμήμα διευθύνσεως του IR (δηλαδή τα 12 λιγότερο σημαντικά bits, από το bit 0 έως το bit 11) να αντιγραφεί στον καταχωρητή MAR της μονάδας μνήμης. Αυτό είναι η πρώτη μικρολειτουργία στην εκτέλεση της εντολής, που θα συμβολίζεται με:

MAR ← IR(0,11)

Στη δεύτερη μικρολειτουργία που αποτελεί την εντολή, γίνεται μία λειτουργία ανάγνωσης στη μνήμη. Με τη λειτουργία αυτή, το περιεχόμενο της διεύθυνσης μνήμης που υποδεικνύει ο MAR μεταφέρεται στον MDR, τον καταχωρητή δεδομένων της μονάδας μνήμης. Η μικρολειτουργία αυτή θα συμβολίζεται σαν:

MDR ← M[MAR] ή απλά **READ**

Στην τρίτη μικρολειτουργία γίνεται τελικά η πρόσθεση μεταξύ των δύο αριθμών. Ο ένας αριθμός λαμβάνεται από τον καταχωρητή δεδομένων MDR και ο άλλος από το συσσωρευτή A. Τα δεδομένα περνούν από τα κυκλώματα της αριθμητικής-λογικής μονάδας που εκτελούν την πρόσθεση και το αποτέλεσμα διοχετεύεται πάλι στο συσσωρευτή A. Η αποθήκευση γίνεται κατευθείαν στον A, μια που αυτός βρίσκεται στο εσωτερικό της αριθμητικής-λογικής μονάδας. Η μικρολειτουργία αυτή θα συμβολίζεται ως:

A ← A + MDR

Μικροπρογραμματισμός

Όπως η εντολή ADA, έτσι και όλες οι υπόλοιπες εντολές του Άβακα και κάθε υπολογιστή μπορούν να αναλυθούν σε μία ακολουθία από μικρολειτουργίες. Οι μικρολειτουργίες που χρειάζονται για να περιγραφούν όλες οι εντολές ενός υπολογιστή δεν είναι πολλές· το πλήθος τους είναι παρόμοιο ή μικρότερο από το πλήθος των εντολών μηχανής του υπολογιστή.

Ένας πιο συστηματικός τρόπος οργάνωσης ενός επεξεργαστή είναι η *μικροπρογραμματιζόμενη αρχιτεκτονική* (microprogrammed architecture). Η βασική της ιδέα είναι να αναλύονται οι εντολές μηχανής σε μικρολειτουργίες, και οι ακολουθίες μικρολειτουργιών για κάθε εντολή να είναι καταχωρημένες σαν προγράμματα σε μία ειδική μνήμη, τη *μνήμη ελέγχου* (control memory). Η μνήμη αυτή βρίσκεται στο εσωτερικό του επεξεργαστή και όχι στη μονάδα μνήμης του υπολογιστή.

Σε κάθε μικρολειτουργία αντιστοιχίζεται μία *μικροεντολή* (microinstruction), και η μνήμη ελέγχου περιέχει μία σειρά μικροεντολών, δηλαδή ένα *μικροπρόγραμμα* (microprogram). Για να εκτελεστεί το μικροπρόγραμμα μίας εντολής από την ΚΜΕ, και να εκτελεστεί η εντολή, είναι απαραίτητοι οι αντίστοχοι καταχωρητές ελέγχου μέσα στην ΚΜΕ, ο *καταχωρητής μικροεντολών* (MIR) και ο *μετρητής μικροπρογράμματος*

(MPC). Οι καταχωρητές αυτοί παίζουν για την ΚΜΕ τον ίδιο ρόλο που παίζουν ο καταχωρητής εντολών IR και ο μετρητής προγράμματος PC για τον υπολογιστή.

Μπορούμε να φανταστούμε έτσι ότι ο επεξεργαστής είναι στην πραγματικότητα ένας «μικρός υπολογιστής» μέσα στον υπολογιστή, που εκτελεί προγράμματα στη δική του γλώσσα μηχανής, τη γλώσσα των μικρολειτουργιών. Κάθε πρόγραμμα στη γλώσσα αυτή είναι μία εντολή μηχανής για τον υπολογιστή.

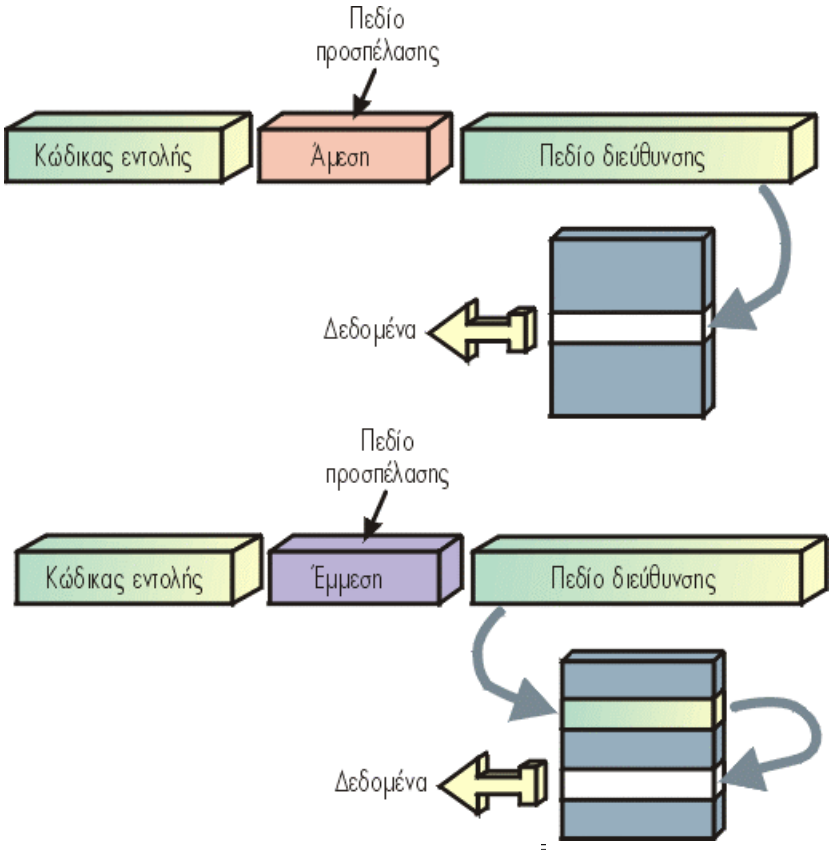
Σε πολλούς υπολογιστές ο κατασκευαστής δίνει τη δυνατότητα στους χρήστες να επέμβουν στο επίπεδο των μικρολειτουργιών και να κατασκευάσουν το δικό τους ρεπερτόριο εντολών μηχανής. Με έμμεσο τρόπο έτσι οι χρήστες μπορούν να επέμβουν «προγραμματιστικά» στην οργάνωση του υλικού. Αυτό το συνδυασμό υλικού και λογισμικού τον ονομάζουμε *υλικολογισμικό* (firmware).

Άλλες αρχιτεκτονικές επεξεργαστών

Ο Άβακας, ο υπολογιστής που είδαμε μέχρι τώρα, είναι πολύ απλός γιατί σκοπός του είναι να καταδείξει τις βασικές αρχές λειτουργίας των επεξεργαστών. Στην πράξη τα πράγματα είναι αρκετά πιο πολύπλοκα.

Σε όλους τους σύγχρονους επεξεργαστές κάθε εντολή αποτελείται από πολλά πεδία, και όχι μόνο από δύο. Εκτός από το πεδίο εντολής και το πεδίο διεύθυνσης, μπορούν να υπάρχουν:

- *Πεδία που προσδιορίζουν τον τρόπο προσπέλασης στη μνήμη.*



Συνήθως το πεδίο διεύθυνσης της εντολής προσδιορίζει τη θέση της μνήμης όπου βρίσκονται κάποια δεδομένα. Ο τρόπος αυτός αναφοράς στη μνήμη ονομάζεται *άμεση αναφορά* (direct access). Το πεδίο διεύθυνσης μπορεί επίσης να προσδιορίζει τη θέση στη μνήμη όπου βρίσκεται η διεύθυνση η οποία τελικά περιέχει τα δεδομένα και πρέπει να χρησιμοποιήσει η εντολή. Αυτή είναι η *έμμεση αναφορά* (indirect access) στη μνήμη και υπάρχει ειδικό πεδίο μέσα στην εντολή που προσδιορίζει αν θα εφαρμοστεί ή όχι. Το πεδίο αυτό μπορεί επίσης να προσδιορίζει ότι τα δεδομένα περιέχονται απευθείας στο τμήμα διεύθυνσης της εντολής, δηλαδή ότι γίνεται *απευθείας αναφορά* (immediate access).

- *Πεδία που προσδιορίζουν τη συνθήκη για τις εντολές άλματος*. Στον Άβακα, η εντολή άλματος υπό συνθήκη ελέγχει αν ο συσσωρευτής Α έχει αρνητική τιμή. Ένα άλμα υπό συνθήκη όμως μπορεί να γίνει ανάλογα με τις τιμές διαφόρων καταχωρητών, με το αποτέλεσμα μίας αριθμητικής ή λογικής πράξης που έγινε προηγουμένως, κλπ. Οι συνθήκες αυτές κωδικοποιούνται μέσα σε ένα πεδίο της εντολής.

- *Πεδία που αναφέρονται σε καταχωρητές της ΚΜΕ*. Όταν η ΚΜΕ περιέχει πολλούς καταχωρητές, οι εντολές μπορούν να αναφέρονται σε οποιονδήποτε από αυτούς.

Μία εντολή της μορφής LDA, μπορεί να μεταφέρει δεδομένα από τη μνήμη σε οποιοδήποτε καταχωρητή και όχι μόνο τον Α. Έτσι υπάρχει ένας κώδικας εντολής, που σημαίνει «μεταφορά από τη μνήμη σε καταχωρητή» και ένα πεδίο εντολής που περιέχει τον «κωδικό» του καταχωρητή στον οποίο θα γραφούν τα δεδομένα.

Υπολογιστές σύνθετου ρεπερτορίου εντολών

Οι *υπολογιστές σύνθετου ρεπερτορίου εντολών* (Complex Instruction Set Computers - CISC) διαθέτουν ένα μεγάλο ρεπερτόριο εντολών (120 έως 350 περίπου), χρησιμοποιούν μικρό αριθμό καταχωρητών (8 έως 24) και επίσης χρησιμοποιούν πολλούς διαφορετικούς τρόπους προσπέλασης στη μνήμη (περισσότερους από 12).

Οι υπολογιστές CISC χρησιμοποιούν συνήθως ένα συνδυασμό μικροπρογραμματιζόμενης και καλωδιωμένης αρχιτεκτονικής. Διαθέτουν λανθάνουσα μνήμη και διαδρόμους μεγάλου πλάτους, κοινούς για δεδομένα και διευθύνσεις.

Το βασικό τους πλεονέκτημα είναι ότι διευκολύνουν ιδιαίτερα τον προγραμματισμό σε γλώσσα μηχανής και οι μεταφραστές για τις γλώσσες προγραμματισμού είναι σχετικά απλοί. Μειονέκτημά τους είναι ότι κάθε εντολή τους απαιτεί αρκετό χρόνο να εκτελεστεί και βέβαια η σχεδίασή τους είναι αρκετά πολύπλοκη.

Οι περισσότεροι προσωπικοί υπολογιστές χρησιμοποιούν επεξεργαστές CISC.

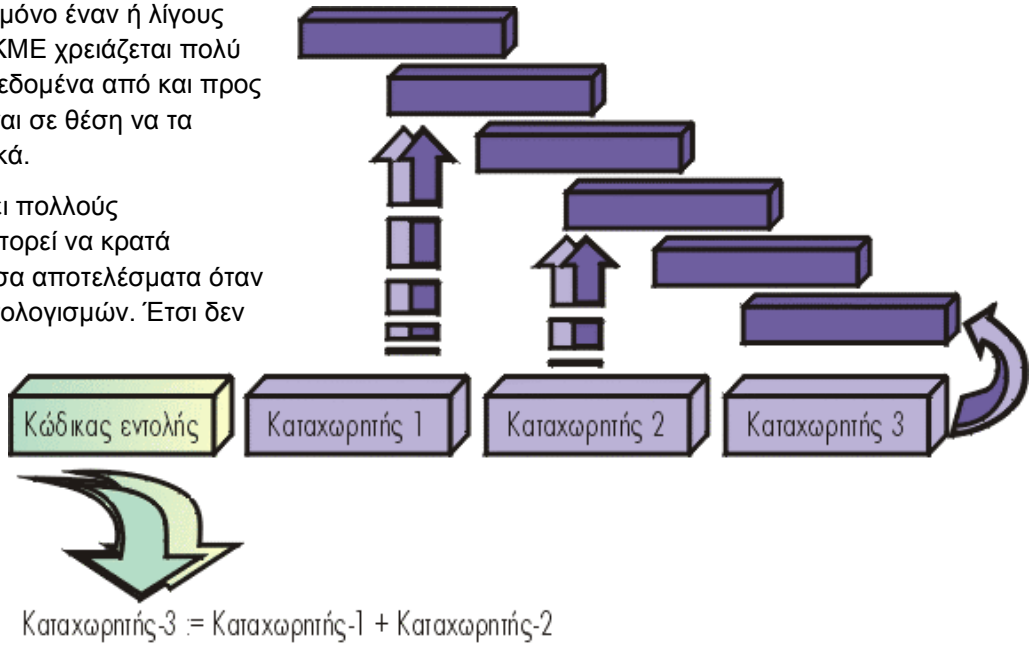
Υπολογιστές μειωμένου ρεπερτορίου εντολών

Με την εξέλιξη των επεξεργαστών CISC, το ρεπερτόριο εντολών τους γινόταν όλο και μεγαλύτερο. Οι σχεδιαστές υπολογιστών παρατήρησαν όμως στατιστικά ότι μόνο το 25% των εντολών ενός υπολογιστή χρησιμοποιούνται πολύ (περίπου το 95% του χρόνου). Το ρεπερτόριο εντολών, απ' ό,τι φαινόταν, θα μπορούσε να μειωθεί δραστικά, απλουστεύοντας έτσι και το υλικό. Οι επιπλέον εντολές θα μπορούσαν να υλοποιηθούν με λογισμικό.

Αυτή είναι η κεντρική ιδέα των *υπολογιστών μειωμένου ρεπερτορίου εντολών* (Reduced Instruction Set Computers, RISC). Το ρεπερτόριο εντολών των υπολογιστών αυτών είναι μικρό, με λιγότερες από 100 εντολές, οπότε το πεδίο κώδικα εντολής δεν καταλαμβάνει μεγάλο χώρο στην εντολή. Έτσι οι εντολές εκτελούνται γρήγορα.

Όταν η ΚΜΕ περιέχει μόνο έναν ή λίγους καταχωρητές, τότε η ΚΜΕ χρειάζεται πολύ συχνά να μεταφέρει δεδομένα από και προς τη μνήμη, γιατί δεν είναι σε θέση να τα αποθηκεύσει εσωτερικά.

Μία ΚΜΕ που περιέχει πολλούς καταχωρητές όμως μπορεί να κρατά εσωτερικά τα ενδιάμεσα αποτελέσματα όταν εκτελεί ακολουθίες υπολογισμών. Έτσι δεν έχει τόσο μεγάλη ανάγκη για ανταλλαγή δεδομένων και προσπέλαση της μνήμης. Αυτό έχει σαν άμεσο αποτέλεσμα την αύξηση της ταχύτητας του υπολογιστή σε σχέση με κάποιον με λίγους καταχωρητές.



> Μία εντολή ενός υπολογιστή RISC είναι αυτή που προσθέτει δύο αριθμούς που είναι αποθηκευμένοι σε δύο καταχωρητές και αποθηκεύει το αποτέλεσμα σε έναν τρίτο καταχωρητή. Για την εντολή αυτή υπάρχουν πολλοί συνδυασμοί εκτέλεσης, αλλά ο κώδικας της εντολής είναι ένας. Σε συμβολική γλώσσα, η μορφή της εντολής θα ήταν:
>
> $$\text{ADD } R_1, R_2, R_3$$
>
> και θα είχε την έννοια $R_3 := R_1 + R_2$.
>
> Αν η ΚΜΕ περιέχει 10 καταχωρητές, τότε για κάθε έναν από τους R_1, R_2, R_3 θα υπάρχουν 10 διαφορετικές επιλογές και οι τελικοί συνδυασμοί θα είναι $10 \times 10 \times 10 = 1000$. Με έναν κώδικα εντολής λοιπόν μπορούν να κωδικοποιηθούν 1000 διαφορετικές εντολές.

Πρακτικά, οι πολλοί καταχωρητές μέσα στην ΚΜΕ συνθέτουν μία πολύ μικρή και γρήγορη μνήμη. Σήμερα οι περισσότεροι επεξεργαστές, κυρίως σε σταθμούς εργασίας, είναι τύπου RISC.

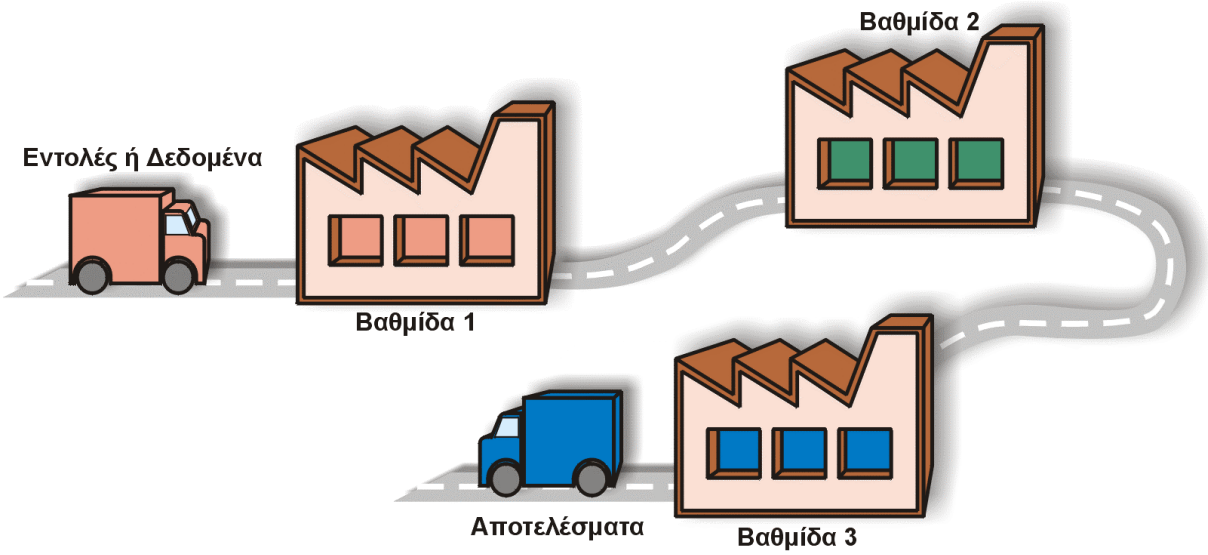
Οι υπολογιστές RISC δε χρησιμοποιούν μικροπρογραμματιζόμενη αρχιτεκτονική εξαιτίας της απλότητας των εντολών τους. Επίσης διαθέτουν ξεχωριστούς διαδρόμους για τα δεδομένα και τις διευθύνσεις τους.

Το μειονέκτημα των υπολογιστών RISC είναι ότι ο προγραμματισμός σε γλώσσα μηχανής γίνεται πιο δύσκολος και απαιτούνται πιο «έξυπνοι» μεταφραστές για τις γλώσσες προγραμματισμού.

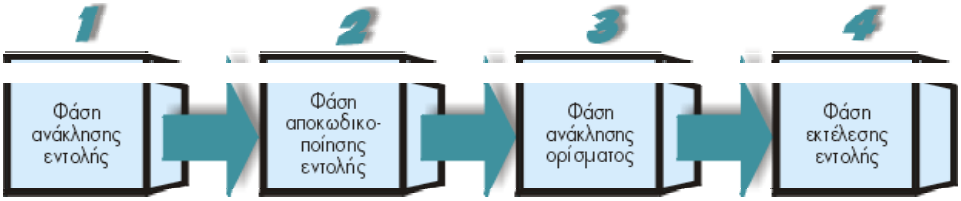
Αρχιτεκτονική αγωγού

Μία αρχιτεκτονική που χρησιμοποιείται σε πολλούς σύγχρονους επεξεργαστές είναι η *αρχιτεκτονική αγωγού* (pipeline architecture).

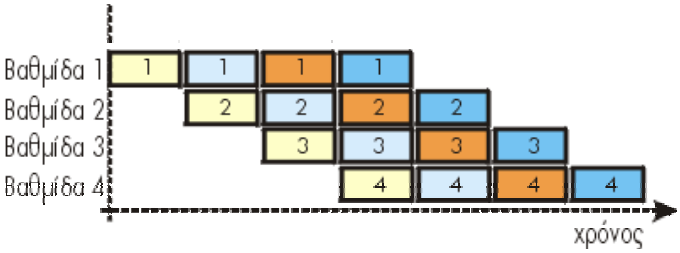
Στους επεξεργαστές αυτούς, ο κύκλος εκτέλεσης μίας εντολής διαιρείται σε πολλές διαδοχικές φάσεις, που ονομάζονται *κύκλοι αγωγού* (pipeline cycles). Μέσα στην ΚΜΕ υπάρχει ένα διακεκριμένο τμήμα που αντιστοιχεί σε κάθε ένα κύκλο και ονομάζεται *βαθμίδα* (stage). Κάθε εντολή ξεκινά από την πρώτη βαθμίδα του αγωγού και περνά από όλες με τη σειρά μέχρι να φθάσει και στην τελευταία. Μόλις μία βαθμίδα τελειώσει την επεξεργασία μίας εντολής, μπορεί αμέσως να προχωρήσει στην επόμενη χωρίς καθυστέρηση.



Ένας υπολογιστής είναι οργανωμένος σε αγωγό τεσσάρων βαθμίδων. Κάθε βαθμίδα εκτελεί μία φάση στην εκτέλεση μίας εντολής, όπως φαίνεται στο σχήμα.



Αν κάθε βαθμίδα της εντολής ολοκληρώνει τη λειτουργία της σε 1 χρονική μονάδα, τότε η εκτέλεση των εντολών γίνεται όπως φαίνεται στο πρώτο χρονικό διάγραμμα εκτέλεσης. Παρατηρούμε ότι η πρώτη εντολή απαιτεί τέσσερις χρονικές μονάδες να ολοκληρωθεί, αλλά από εκεί και στο εξής σε κάθε χρονική μονάδα ολοκληρώνεται μία εντολή. Έτσι μακροπρόθεσμα μπορούμε να πούμε ότι ο υπολογιστής αυτός εκτελεί μία εντολή ανά χρονική μονάδα.



Εκτέλεση εντολών στον υπολογιστή με αγωγό: κατά μέσο όρο ολοκληρώνεται μία εντολή σε κάθε χρονική μονάδα

Αν αντιπαραβάλουμε τον υπολογιστή αυτόν με κάποιον που δεν έχει αγωγό αλλά εκτελεί τις εντολές μία μία βλέπουμε ότι είναι τέσσερις φορές πιο γρήγορος, αφού ο υπολογιστής χωρίς αγωγό χρειάζεται τέσσερις χρονικές μονάδες ανά εντολή, μία ανά φάση εκτέλεσης.

Εκτέλεση εντολών στον υπολογιστή χωρίς αγωγό: κατά μέσο όρο ολοκληρώνεται μία εντολή κάθε τέσσερις χρονικές μονάδες

Μία ιδιαιτερότητα της αρχιτεκτονικής αγωγού είναι ο τρόπος που εκτελούνται οι εντολές άλματος υπό συνθήκη. Το αν θα εκτελεστεί ένα τέτοιο άλμα ή όχι, συνήθως εξαρτάται από το αποτέλεσμα μίας αριθμητικής ή λογικής πράξης, η οποία μπορεί να μην έχει ακόμα ολοκληρωθεί όταν η εντολή άλματος «εισαχθεί» στον αγωγό. Έτσι λοιπόν ο επεξεργαστής πρέπει να λάβει υπόψη του συγχρόνως και τα δύο «σενάρια» εκτέλεσης του προγράμματος. Όταν ολοκληρωθεί η λειτουργία που επηρεάζει την εκτέλεση του άλματος, ο επεξεργαστής θα επιλέξει το κατάλληλο «σενάριο» και θα το συνεχίσει.

Εκτός από τους αγωγούς που αναλαμβάνουν ολόκληρη την εκτέλεση μίας εντολής, υπάρχουν και οι *αριθμητικοί αγωγοί*, οι οποίοι χρησιμοποιούνται για το γρηγορότερο υπολογισμό αριθμητικών εκφράσεων.

Μία αρχιτεκτονική παρόμοια με την αρχιτεκτονική αγωγού είναι η *υπερβαθμωτή* (superscalar) αρχιτεκτονική. Σε αυτήν, ο επεξεργαστής μπορεί να εκτελέσει παράλληλα πολλές, ανεξάρτητες μεταξύ τους εντολές μηχανής, ενώ εντολές που επηρεάζουν η μία την άλλη εκτελούνται ακολουθιακά.

Υπάρχουν ακόμα διάφορες άλλες αρχιτεκτονικές υπολογιστών, όπως:

- **Οι παράλληλοι υπολογιστές**. Στο μοντέλο αυτό, πολλοί επεξεργαστές συνεργάζονται και συγχρονίζονται για να εκτελέσουν ένα πρόγραμμα γραμμένο έτσι ώστε να μοιράζει το υπολογιστικό φορτίο ανάμεσα στις διάφορες ΚΜΕ. Οι επεξεργαστές αυτοί μπορούν να έχουν κοινή μνήμη, οπότε ονομάζονται *πολυεπεξεργαστές* (multiprocessors) ή να έχει καθένας τη δική του μνήμη, οπότε μιλάμε για *πολυυπολογιστές* (multicomputers).
- **Οι μη συμβατικοί υπολογιστές**. Αυτοί δεν ακολουθούν το κλασικό μοντέλο του Von Neumann όπως ο Άβακας, όπου το πρόγραμμα είναι αποθηκευμένο στη μνήμη και υπάρχει ο μετρητής προγράμματος. Τέτοιοι υπολογιστές είναι οι *μηχανές ροής δεδομένων* (data flow machines), και οι *μηχανές αναγωγής* (reduction machines).
- **Οι υπολογιστές ειδικού σκοπού**, όπως οι *συστολικές διατάξεις* (systolic arrays), οι *κυτταρικές διατάξεις* (cellular arrays), οι ειδικοί επεξεργαστές για επεξεργασία σήματος (DSP) κλπ.
- **Οι *υπολογιστές μεγάλου μήκους λέξης* ή *πολύ μεγάλου μήκους λέξης*** (long instruction word computers, LIW ή very long instruction word computers, VLIW). Σε αυτούς το μήκος λέξης είναι πολύ μεγαλύτερο από ό,τι στους περισσότερους υπολογιστές, της τάξης των εκατοντάδων bits. Κάθε εντολή περιλαμβάνει την ταυτόχρονη εκτέλεση πολλών λειτουργιών.

Ανακεφαλαίωση

Σε πολλούς σύγχρονους επεξεργαστές οι εντολές σε γλώσσα μηχανής αποτελούν με τη σειρά τους προγράμματα που αποτελούνται από στοιχειώδεις λειτουργίες του επεξεργαστή, τις μικροεντολές.
Στους σύγχρονους επεξεργαστές συναντάμε μία ποικιλία αρχιτεκτονικών, όπως είναι η αρχιτεκτονική μειωμένου ρεπερτορίου εντολών, η αρχιτεκτονική αγωγού, η υπερβαθμωτή αρχιτεκτονική κλπ.

Γλωσσάριο όρων

Άμεση Αναφορά	*Direct Access*
Απευθείας Αναφορά	*Immediate Access*
Αρχιτεκτονική Αγωγού	*Pipeline Architecture*
Βαθμίδα	*Stage*
Έμμεση Αναφορά	*Indirect Access*
Καταχωρητής Μικροεντολών	*MIR*
Κύκλοι Αγωγού	*Pipeline Cycles*
Κυτταρικές Διατάξεις	*Cellular Arrays*
Μετρητής Μικροπρογράμματος	*MPC*
Μηχανές Αναγωγής	*Reduction Machines*
Μηχανές Ροής Δεδομένων	*Data Flow Machines*
Μικροεντολή	*Microinstruction*
Μικρολειτουργία	*Microoperation*
Μικροπρόγραμμα	*Microprogram*
Μικροπρογραμματιζόμενη Αρχιτεκτονική	*Microprogrammed Architecture*
Μνήμη Ελέγχου	*Control Memory*
Πολυεπεξεργαστές	*Multiprocessors*
Πολυυπολογιστές	*Multicomputers*
Συστολικές Διατάξεις	*Systolic Arrays*
Υλικολογισμικό	*Firmware*
Υπερβαθμωτή Αρχιτεκτονική	*Superscalar Architecture*
Υπολογιστής Μεγάλου Μήκους Λέξης	*Long Instruction Word - LIW*
Υπολογιστής Μειωμένου Ρεπερτορίου Εντολών	*Reduced Instruction Set Computer - RISC*
Υπολογιστής Σύνθετου Ρεπερτορίου Εντολών	*Complex Instruction Set Computer - CISC*

Ερωτήσεις

? Τι είναι η μικρολειτουργία και το μικροπρόγραμμα;

? Ποιο είναι το χαρακτηριστικό των υπολογιστών μειωμένου ρεπερτορίου εντολών;

? Πώς λειτουργούν οι υπολογιστές με αρχιτεκτονική αγωγού;

Τι μάθαμε σε αυτό το κεφάλαιο

- Ένας υπολογιστής περιλαμβάνει την αριθμητική-λογική μονάδα, τη μονάδα ελέγχου, τη μονάδα μνήμης και τη μονάδα εισόδου/εξόδου. Οι δύο πρώτες αποτελούν την Κεντρική Μονάδα Επεξεργασίας.
- Η εκτέλεση μίας εντολής μηχανής από έναν υπολογιστή περιλαμβάνει τη φάση ανάκλησης της εντολής και τη φάση εκτέλεσης.
- Ο εκπαιδευτικός υπολογιστής Άβακας διαθέτει ένα βασικό καταχωρητή, ο οποίος είναι ο συσσωρευτής Α και 13 εντολές μηχανής.
- Επειδή ο προγραμματισμός σε γλώσσα μηχανής είναι δύσκολος, γράφουμε τις εντολές του προγράμματος σε συμβολική γλώσσα, δηλαδή με συμβολικά ονόματα.
- Τα προγράμματα μετατρέπονται από μία γλώσσα προγραμματισμού υψηλού επιπέδου σε γλώσσα μηχανής από ειδικά προγράμματα: τους μεταφραστές και τους διερμηνείς.
- Σε πολλούς σύγχρονους επεξεργαστές οι εντολές σε γλώσσα μηχανής αποτελούν με τη σειρά τους προγράμματα που αποτελούνται από στοιχειώδεις λειτουργίες του επεξεργαστή, τις μικροεντολές.

Βιβλιογραφία – Πηγές

- Luce T., Αρχιτεκτονική Υπολογιστών, Τζιόλας, 1991.
- Ματζάκος Π. , Γλώσσα Assembly, 1991.
- Παπακωσταντίνου Γ., Π. Τσανάκα, Γ. Φραγκάκη, Αρχιτεκτονική Υπολογιστών, Συμμετρία, 1987.
- Σκορδαλάκη Ε., Εισαγωγή στους μεταγλωττιστές, Συμμετρία,1993.
- Hayes J., Computer Architecture and Organization, McGraw-Hill, 1988.
- Hutchinson S., Sawyer S., Computers and Information Systems, Irwin, 1996.
- Hwang K., Advanced Computer Architecture, McGraw-Hill, 1993.
- Patterson D., Hennessy J. Computer Organization & Design, The Hardware/Software Interface, Morgan Kaufmann, 1994.
- Vermaat M., Walker T., Hall T., Repede J., Discovering Computer, Shelly Cashman Series, 1997.

Κεφάλαιο

Οργάνωση Μνήμης

Σκοπός του κεφαλαίου αυτού είναι να περιγράψει τον τρόπο που διαχειρίζονται οι υπολογιστές τη μνήμη που περιέχουν, ώστε αυτή να κατανέμεται δίκαια και με ασφάλεια μεταξύ των προγραμμάτων, και ο υπολογιστής να λειτουργεί όσο πιο αποδοτικά είναι δυνατό.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις τα τμήματα του συστήματος μνήμης και τα χαρακτηριστικά τους.
- Να εξηγείς τον τρόπο ανάγνωσης και εγγραφής στην κύρια μνήμη.
- Να περιγράφεις τα επίπεδα ιεραρχίας της μνήμης και τα χαρακτηριστικά τους.
- Να καταρτίσεις την κύρια μνήμη για ένα ορισμένο υπολογιστή και να χρησιμοποιείς την τεχνική της διαφύλλωσης.
- Να περιγράφεις τον τρόπο λειτουργίας της λανθάνουσας μνήμης και να υπολογίζεις την επιτάχυνση που προσφέρει στα προγράμματα.

Μαθήματα

4.1 Τεχνολογία και Χαρακτηριστικά Μνημών

4.2 Ιεραρχία Μνήμης

4.3 Σχεδίαση και Οργάνωση Κύριας Μνήμης

4.4 Λανθάνουσα Μνήμη

Μάθημα 4.1 Τεχνολογία και Χαρακτηριστικά Μνημών

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τα κύρια χαρακτηριστικά του συστήματος μνήμης για έναν υπολογιστή και τις βασικές λειτουργίες της κύριας μνήμης.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις τα βασικά χαρακτηριστικά των μνημών*
- *Να απαριθμείς τους βασικούς στόχους των σχεδιαστών ενός συστήματος μνήμης*
- *Να περιγράφεις τη διαδικασία ανάγνωσης και εγγραφής της κύριας μνήμης*
- *Να προσδιορίζεις το χώρο διευθύνσεων της μνήμης*

Το *υποσύστημα της μνήμης* σε έναν υπολογιστή περιλαμβάνει όλες εκείνες τις μονάδες που έχουν τη δυνατότητα να αποθηκεύουν δυαδικές πληροφορίες και να επιτρέπουν την αξιόπιστη ανάκτησή τους οποτεδήποτε αυτό ζητηθεί από τον επεξεργαστή.

Η αποθήκευση των πληροφοριών σε μία μονάδα μνήμης γίνεται με τη λειτουργία της *εγγραφής* (write) στη μνήμη και η ανάκτησή τους γίνεται με τη λειτουργία της *ανάγνωσης* (read) από τη μνήμη. Και οι δύο λειτουργίες ενεργοποιούνται μέσω των αντίστοιχων εντολών του επεξεργαστή.

Οι λειτουργίες της εγγραφής και της ανάγνωσης σε κάθε μονάδα μνήμης, για να λειτουργεί η μονάδα ικανοποιητικά, πρέπει να εκτελούνται:

(1) **Αξιόπιστα,** δηλαδή χωρίς σφάλματα. Όταν εκτελείται μια λειτουργία ανάγνωσης στη μνήμη, τα δεδομένα που θα διαβαστούν πρέπει να είναι *ακριβώς* αυτά που είχαν εγγραφεί την τελευταία φορά σε αυτή τη θέση.

(2) **Ταχύτατα**, όσο βέβαια επιτρέπουν οι φυσικοί περιορισμοί της μονάδας μνήμης.

(3) **Με το ελάχιστο δυνατό κόστος**, ώστε να είναι εφικτή η αποθήκευση μεγάλου όγκου δεδομένων χωρίς υπέρμετρη οικονομική επιβάρυνση.

Οι πληροφορίες που είναι αποθηκευμένες σε μια μονάδα μνήμης είναι συνήθως οργανωμένες σε ομάδες δυαδικών ψηφίων. Κάθε ομάδα χαρακτηρίζεται από τα *περιεχόμενά* της και από τη *διεύθυνσή* της. Κάθε προσπέλαση (ανάγνωση ή εγγραφή) στη μνήμη γίνεται για τη μεταφορά μιας ενιαίας ομάδας πληροφοριών, με βάση τη διεύθυνση τής ομάδας. Οι συνηθέστεροι τύποι ομάδων είναι η λέξη και η ενότητα.

Η *λέξη* (word) συνήθως αποτελείται από 1-8 bytes (δηλαδή, 8-64 bits) και χρησιμοποιείται για την προσπέλαση στη μονάδα της κύριας μνήμης του υπολογιστή.

Η *ενότητα* (block ή frame), η οποία συνήθως περιλαμβάνει έναν μεγάλο αριθμό από λέξεις, χρησιμοποιείται συχνά ως βασική μονάδα ανταλλαγής πληροφορίας με τις περιφερειακές συσκευές. Έτσι η ανάγνωση ή η εγγραφή στο σκληρό δίσκο δεν μπορεί να γίνει για μία μόνο λέξη, αλλά για ολόκληρη την ενότητα στην οποία αυτή περιλαμβάνεται.

Μονάδα	Κόστος (δρχ/bit)	Ταχύτητα	Χωρητικότητα (Mbytes)
Η μνήμη ενός προσωπικού υπολογιστή το 1980	0,1	120 ns	0,2
Η μνήμη ενός προσωπικού υπολογιστή το 1999	10^{-4}	60 ns	64
Ένας σκληρός δίσκος για εξυπηρετητή (server) το 1999	10^{-5}	10 ms	10000

Μερικά παραδείγματα μονάδων μνήμης, παλιών και νεότερων, και των χαρακτηριστικών τους.

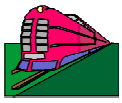
K	Kilo	2^{10}
M	Mega	2^{20}
G	Giga	2^{30}
T	Tera	2^{40}
m	milli	10^{-3}
μ	μικρο	10^{-6}
n	nano	10^{-9}

Οι πολλαπλασιαστικοί παράγοντες που χρησιμοποιούνται περισσότερο στο χώρο των υπολογιστών

Σε κάθε υπολογιστή υπάρχουν διάφορες μορφές μνήμης, που διαφέρουν μεταξύ τους ως προς τα ποσοτικά χαρακτηριστικά τους, όπως:

- ⇨ *Το κόστος τους*, σε δρχ ανά bit αποθηκευμένης πληροφορίας
- ⇨ *Το χρόνο προσπέλασης* (access time). Αυτός είναι ο χρόνος που απαιτείται για την ανάγνωση ή εγγραφή στη μνήμη μίας μονάδας πληροφορίας. Ο χρόνος προσπέλασης σήμερα μετράται σε εκατομμυριοστά του δευτερολέπτου (ns).
- ⇨ *Τη χωρητικότητά της* (storage capacity), δηλαδή τον συνολικό όγκο πληροφοριών που μπορούν να αποθηκευθούν στη μονάδα αυτή. Η χωρητικότητα μιας μονάδας μνήμης μετράται σε εκατομμύρια bytes ή MBytes*.

Οι σχεδιαστές ενός συστήματος μνήμης επιδιώκουν:



Να μεγιστοποιήσουν την ταχύτητα μεταφοράς των πληροφοριών από και προς τη μονάδα αποθήκευσης. Έτσι ελαχιστοποιείται ο χρόνος προσπέλασης στη μνήμη.

Να ελαχιστοποιήσουν το μέσο κόστος αποθήκευσης, ώστε μονάδες μνήμης με μεγάλη χωρητικότητα να έχουν προσιτή τιμή.

Η διαδικασία της μεταφοράς πληροφοριών μεταξύ των διαφόρων μονάδων μνήμης να είναι αυτοματοποιημένη. Έτσι, όταν οι προγραμματιστές γράφουν προγράμματα, τα οποία χρησιμοποιούν τις μονάδες μνήμης, δε χρειάζεται να καταβάλουν προσπάθεια στον τομέα αυτό.

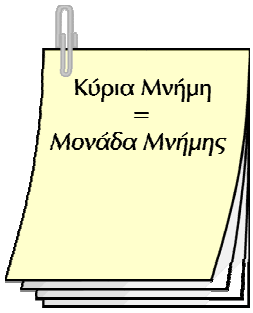
* Στο χώρο των υπολογιστών, το πρόθεμα Mega συμβολίζει έναν πολλαπλασιαστικό παράγοντα με τιμή 2^{20} = 1.048.576, και όχι τον συνηθισμένο παράγοντα 10^6 = 1.000.000. Επειδή όμως οι δύο τιμές είναι παραπλήσιες, πολλές φορές λέμε «εκατομμύρια» και εννοούμε 1.048.576.

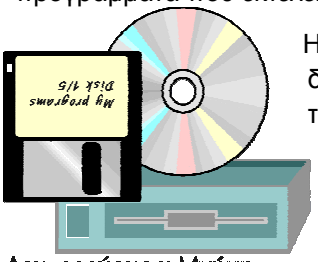
Να παρέχουν αποτελεσματικούς μηχανισμούς που θα προστατεύουν τα αποθηκευμένα δεδομένα. Ο κίνδυνος για αυτά προέρχεται από σφάλματα ή ανεπίτρεπτες ενέργειες των προγραμμάτων, που είναι δυνατόν να οδηγήσουν στην καταστροφή των δεδομένων ή σε αθέμιτη αντιγραφή τους.

Κύρια και δευτερεύουσα μνήμη

Η μνήμη ενός υπολογιστή διακρίνεται συνήθως στην *κύρια μνήμη* (main memory) και τη *δευτερεύουσα μνήμη* (secondary memory).

Η κύρια μνήμη είναι αυτή που είδαμε στο μάθημα 3.1 και μπορεί να προσπελαστεί απ' ευθείας από την ΚΜΕ, χωρίς να παρεμβληθεί η μονάδα εισόδου/εξόδου. Ένα βασικό της χαρακτηριστικό είναι ότι τα περιεχόμενά της χάνονται, όταν σταματήσει η λειτουργία του υπολογιστή. Στην κύρια μνήμη αποθηκεύονται από την ΚΜΕ τα προγράμματα που εκτελεί, καθώς και τα δεδομένα των προγραμμάτων αυτών.



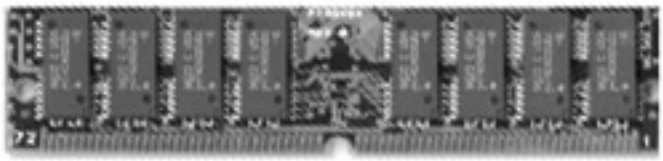


Δευτερεύουσα Μνήμη = Περιφερειακές Συσκευές

Η δευτερεύουσα ή βοηθητική μνήμη απαρτίζεται από διάφορες περιφερειακές συσκευές. Σ' αυτήν αποθηκεύονται τα δεδομένα που δεν είναι άμεσα απαραίτητα στην ΚΜΕ, τα οποία διατηρούνται αναλλοίωτα και μετά τη διακοπή λειτουργίας του υπολογιστή. Η δευτερεύουσα μνήμη αποτελείται συνήθως από μαγνητικούς δίσκους, οπτικούς δίσκους (CD-ROM) και μαγνητικές ταινίες.

Μνήμη άμεσης προσπέλασης

Η κύρια μνήμη του υπολογιστή είναι *μνήμη άμεσης προσπέλασης* (Random Access Memory). Σε μία τέτοια μνήμη, ο χρόνος που χρειάζεται για να διαβάσουμε ή να γράψουμε μία πληροφορία στη μνήμη είναι πάντα ο ίδιος, ανεξάρτητα από τη συγκεκριμένη διεύθυνση της πληροφορίας αυτής.



Μια μονάδα μνήμης άμεσης προσπέλασης πραγματοποιείται συνήθως με ένα ολοκληρωμένο κύκλωμα, το οποίο επικοινωνεί με τον υπολογιστή με ηλεκτρικά σήματα που περνούν από τους ακροδέκτες του. Υπάρχουν τέτοια σήματα για τον προσδιορισμό μιας διεύθυνσης, για την ανταλλαγή των πληροφοριών που αποθηκεύει η μνήμη και για τον προσδιορισμό της λειτουργίας που θα επιτελέσει.

Οι δύο βασικές τεχνολογίες που χρησιμοποιούνται σήμερα για την κύρια μνήμη είναι (α) οι στατικές μνήμες (SRAM), με χρόνο προσπέλασης 10-20 ns και (β) οι δυναμικές μνήμες (DRAM), με χρόνο προσπέλασης 90-120 ns*. Το κόστος των δυναμικών μνημών είναι περίπου το 25% αυτού των στατικών μνημών.

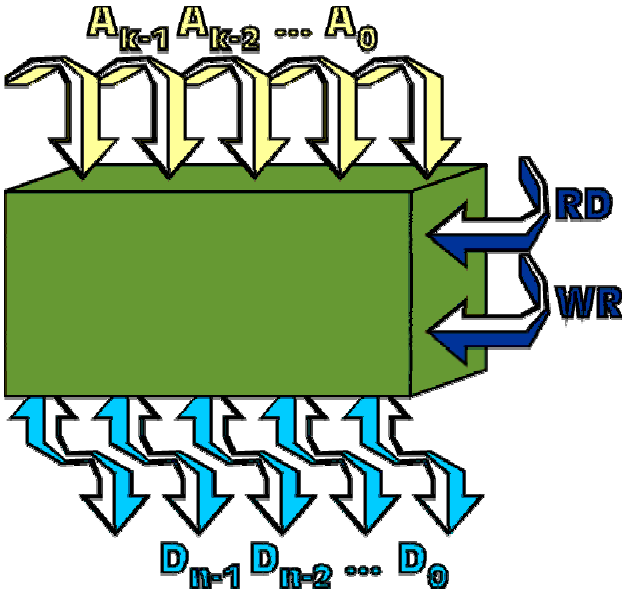
Αν το πλήθος των λέξεων σε μία μονάδα μνήμης είναι δύναμη του 2, π.χ. 2^K, τότε οι διευθύνσεις των λέξεων χρειάζονται *ακριβώς* K bits για να παρασταθούν.

Μπορούμε να φανταζόμαστε μια μονάδα μνήμης άμεσης προσπέλασης σαν ένα πίνακα από λέξεις, κάθε μία από τις οποίες αντιστοιχεί σε μία διεύθυνση. Αν η μονάδα περιέχει 2^K λέξεις, οι διευθύνσεις τους θα είναι οι αριθμοί από 0 έως 2^K-1, και θα απαιτούν κ bits για να παρασταθούν.

* Οι τιμές αυτές, οι οποίες βελτιώνονται συνεχώς, είναι ενδεικτικές για το 1999.

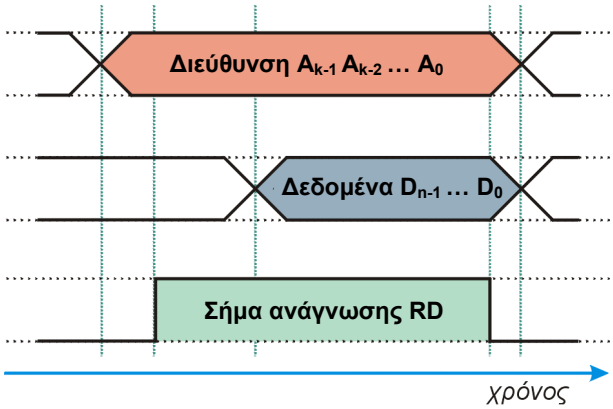
Έτσι λοιπόν το ολοκληρωμένο κύκλωμα της μνήμης επικοινωνεί με τις υπόλοιπες μονάδες του υπολογιστή με:

- ✓ Τις *γραμμές διεύθυνσης*, που είναι k το πλήθος, και έχουν τα ονόματα A_{k-1}-A_0.
- ✓ Τις *γραμμές δεδομένων*, που είναι n το πλήθος (n είναι το μέγεθος σε bits της λέξης για τη μνήμη) με ονόματα D_{n-1}-D_0.
- ✓ Μια γραμμή για την ενεργοποίηση της εγγραφής στη μνήμη, με το όνομα WR (από το Write, γράφω)
- ✓ Μια γραμμή για την ενεργοποίηση της ανάγνωσης από τη μνήμη, με το όνομα RD (από το Read, διαβάζω)

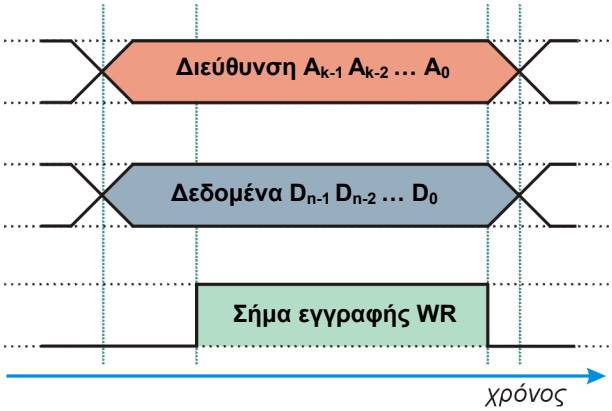


Για να γίνει μια λειτουργία ανάγνωσης από τη μονάδα της μνήμης, οι γραμμές διεύθυνσης (A_{k-1}…A_0) μεταφέρουν τη δυαδική διεύθυνση της λέξης που θα διαβαστεί. Στη συνέχεια ο επεξεργαστής ενεργοποιεί το σήμα RD, και μετά από μια μικρή καθυστέρηση οι γραμμές δεδομένων (D_{n-1}…D_0) ενεργοποιούνται με τα δεδομένα της λέξης.

Με παρόμοιο τρόπο, για να γίνει μια εγγραφή στη μονάδα της μνήμης, οι γραμμές *διεύθυνσης* μεταφέρουν τη διεύθυνση της λέξης που θα γραφεί και οι γραμμές δεδομένων τα περιεχόμενά της. Κατόπιν ο επεξεργαστής ενεργοποιεί το σήμα WR και μετά από ένα μικρό χρονικό διάστημα η λέξη έχει εγγραφεί στη μνήμη.



Χρονική εξέλιξη των σημάτων κατά την ανάγνωση από τη μνήμη



Χρονική εξέλιξη των σημάτων κατά την εγγραφή στη μνήμη

Ο χώρος διευθύνσεων του επεξεργαστή

Ο επεξεργαστής του υπολογιστή μπορεί να αναφερθεί άμεσα στις λέξεις της μνήμης οι οποίες περιγράφονται από μια διεύθυνση μεταξύ 0 και 2^k-1, αφού οι γραμμές διεύθυνσης είναι k. Το σύνολο των διευθύνσεων αυτών αποτελεί το *χώρο διευθύνσεων* (address space) του επεξεργαστή.

Οι σύγχρονοι επεξεργαστές χαρακτηρίζονται από σχετικά μεγάλο χώρο διευθύνσεων.

> Οι επεξεργαστές Intel της οικογένειας Pentium διαθέτουν 32 γραμμές διεύθυνσης, οπότε έχουν χώρο διευθύνσεων 2^{32} bytes = 4×2^{30} bytes = 4 GB.

Η αλματώδης αύξηση των απαιτήσεων των εφαρμογών σε μνήμη έχει απαξιώσει πρόωρα πολλούς τύπους υπολογιστών, επειδή αυτοί μπορούσαν να αναφερθούν σε μικρό χώρο διευθύνσεων, και έτσι αποδείχθηκαν ανεπαρκείς για νεότερα προγράμματα και λειτουργικά συστήματα.

Άλλοι τύποι μνήμης

Η δευτερεύουσα μνήμη χρησιμοποιείται για τη μόνιμη και οικονομικότερη αποθήκευση μεγάλου όγκου πληροφοριών.

Κύριο χαρακτηριστικό των περιφερειακών μονάδων που αποτελούν τη δευτερεύουσα μνήμη είναι ο συνδυασμός ηλεκτρομηχανικών διατάξεων για τη μαγνητική ή οπτική αποθήκευση και ανάκτηση μεγάλου όγκου πληροφοριών. Το μηχανικό μέρος αυτών των συσκευών τις καθιστά βραδύτερες σε σχέση με την κύρια μνήμη, της οποίας η λειτουργία είναι καθαρά ηλεκτρονική.

Επειδή η διαδικασία του εντοπισμού και της ανάκτησης των δεδομένων στις μονάδες αυτές είναι σχετικά χρονοβόρα, οι λειτουργίες ανάγνωσης και εγγραφής γίνονται για μεγαλύτερες ενότητες δεδομένων από ό,τι στην κύρια μνήμη.

> Σε ένα σκληρό δίσκο η ενότητα είναι συχνά 512 bytes, δηλαδή πολύ μεγαλύτερη από τη λέξη του επεξεργαστή, που σε πολλούς επεξεργαστές είναι μόνο 1 byte.
>
> Για να διαβάσουμε από το δίσκο αυτό 2000 συνεχόμενα bytes θα χρειαστούν 4 αναγνώσεις. Αν όμως η μονάδα διαβάζει ενότητες των 8 bytes, θα απαιτούνταν 250 λειτουργίες ανάγνωσης και 50 φορές περισσότερος χρόνος!

Ας σημειώσουμε ότι η δευτερεύουσα μνήμη δεν είναι άμεσα προσπελάσιμη από τον επεξεργαστή· τα περιεχόμενά της πρέπει πρώτα να μεταφερθούν στην κύρια μνήμη με τη βοήθεια της μονάδας εισόδου-εξόδου, και μετά να χρησιμοποιηθούν από τον επεξεργαστή.

Ανακεφαλαίωση

Η μνήμη του υπολογιστή συνήθως οργανώνεται σε ομάδες δυαδικών ψηφίων, τις λέξεις. Κάθε λέξη έχει μία διεύθυνση, η οποία είναι ένας αριθμός. Η κύρια μνήμη του υπολογιστή συνήθως είναι μνήμη άμεσης προσπέλασης, όπου κάθε λέξη απαιτεί τον ίδιο χρόνο για να διαβαστεί ή να γραφεί. Αυτό γίνεται ενεργοποιώντας, με τη διεύθυνση της λέξης, τις κατάλληλες γραμμές εξόδου στο ολοκληρωμένο κύκλωμα της μνήμης και διαβάζοντας ή γράφοντας τα δεδομένα στις αντίστοιχες γραμμές-ακροδέκτες του κυκλώματος.

Γλωσσάριο όρων

Ανάγνωση	*Read*
Γραμμές Δεδομένων	*Data Lines*
Γραμμές Διεύθυνσης	*Address Lines*
Δευτερεύουσα Μνήμη	*Secondary Memory*
Εγγραφή	*Write*
Ενότητα	*Block ή Frame*
Κύρια Μνήμη	*Main Memory*
Λέξη	*Word*
Μνήμη Άμεσης Προσπέλασης	*Random Access Memory*
Χρόνος Προσπέλασης	*Access Time*
Χωρητικότητα	*Storage Capacity*
Χώρος Διευθύνσεων	*Address Space*

Ερωτήσεις

? Τι είναι η λέξη μνήμης;

? Τι είναι η ενότητα μνήμης;

? Τι είναι η κύρια και τι η δευτερεύουσα μνήμη;

? Πώς επικοινωνεί το ολοκληρωμένο κύκλωμα της μνήμης με τον υπόλοιπο υπολογιστή;

? Τι είναι ο χώρος διευθύνσεων ενός επεξεργαστή;

Μάθημα 4.2 Ιεραρχία Μνήμης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει την ιεραρχική οργάνωση της μνήμης στους υπολογιστές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

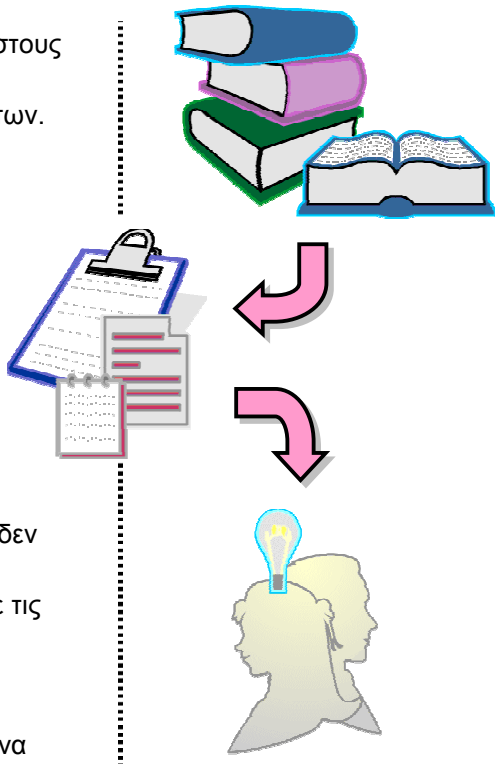
- *Να εξηγείς την τοπικότητα της αναφοράς των προγραμμάτων*
- *Να απαριθμείς τα διάφορα επίπεδα στην ιεραρχία της μνήμης*
- *Να περιγράφεις τις ιδιότητες του κάθε επιπέδου*

Η τοπικότητα αναφοράς για τα προγράμματα και τα δεδομένα

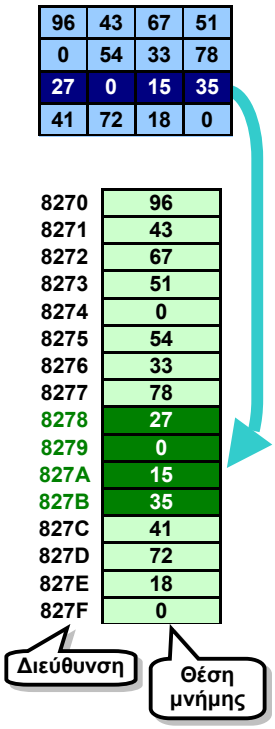
Η σπουδαιότερη ιδιότητα που χαρακτηρίζει την εκτέλεση των προγραμμάτων των υπολογιστών είναι η *τοπικότητα της αναφοράς* (locality of reference). Όταν ένα πρόγραμμα εκτελείται από την ΚΜΕ, η επόμενη εντολή που θα ανακληθεί κάθε φορά από τη μνήμη και θα εκτελεστεί στην ΚΜΕ θα ανήκει πιθανότατα σε κάποια γειτονική θέση σε σχέση με αυτή που εκτελέστηκε αμέσως προηγουμένως.

Στην καθημερινή μας ζωή η τοπικότητα της αναφοράς εμφανίζεται στους αριθμούς τηλεφώνου που χρησιμοποιούμε, γιατί οι περισσότερες τηλεφωνικές κλήσεις μας γίνονται σε περιορισμένο αριθμό προσώπων.

Για να εκμεταλλευθούμε την τοπικότητα αυτή, καταχωρίζουμε τους αριθμούς που μας ενδιαφέρουν σε προσωπικά ευρετήρια ή σημειωματάρια, αντί να χρησιμοποιούμε τους επίσημους τηλεφωνικούς καταλόγους. Επίσης συνηθίζουμε να απομνημονεύουμε μερικούς τηλεφωνικούς αριθμούς που χρησιμοποιούμε πάρα πολύ τακτικά, ώστε η ανάκλησή τους να γίνεται ακόμη πιο γρήγορα. Έτσι, διευκολυνόμαστε στις περισσότερες επικοινωνίες μας, αποφεύγοντας την αναζήτηση στοιχείων στους ογκώδεις καταλόγους, στους οποίους καταφεύγουμε μόνο σε σπάνιες περιπτώσεις.



Λόγω του περιορισμένου τους μεγέθους, τα προσωπικά ευρετήρια δεν μπορούν να καλύψουν πάντα τους αριθμούς τηλεφώνων που χρησιμοποιούμε πολύ συχνά. Αναγκαζόμαστε τότε να διαγράψουμε τις παλαιότερες ή λιγότερο χρησιμοποιούμενες εγγραφές, για να τις αντικαταστήσουμε με νέες, αναμένοντας ότι αυτές θα ζητηθούν συχνότερα στο άμεσο μέλλον. Κάτι παρόμοιο γίνεται και με τους απομνημονευμένους αριθμούς, αφού ξεχνάμε όσους έχουμε καιρό να καλέσουμε, αλλά θυμόμαστε αυτούς που καλέσαμε πρόσφατα.



Στην περίπτωση των προγραμμάτων, η τοπικότητα της αναφοράς είναι πάρα πολύ συχνή. Εκτός από τις εντολές, ισχύει και για τις αναφορές προς τα δεδομένα που είναι αποθηκευμένα στη μνήμη.

> Ένα πρόγραμμα μαθηματικών, που εκτελεί πράξεις σε πίνακες, αποθηκεύει τα στοιχεία των πινάκων κατά γραμμές στη μνήμη. Ο πίνακας του σχήματος, που έχει διαστάσεις 4×4, αποθηκεύεται ξεκινώντας από τη διεύθυνση $8270_{(16)}$ της μνήμης.
>
> Όπως βλέπουμε και στο σχήμα, τα περιεχόμενα του πίνακα αποθηκεύονται σε συνεχόμενες θέσεις της μνήμης. Όταν π.χ. το πρόγραμμα εκτελεί μια πράξη που χρησιμοποιεί τον πίνακα κατά γραμμές, θα προσπελάσει με τη σειρά όλες τις θέσεις μνήμης που του αναλογούν, ξεκινώντας από τη διεύθυνση 8270, και συνεχίζοντας μέχρι την 827F.

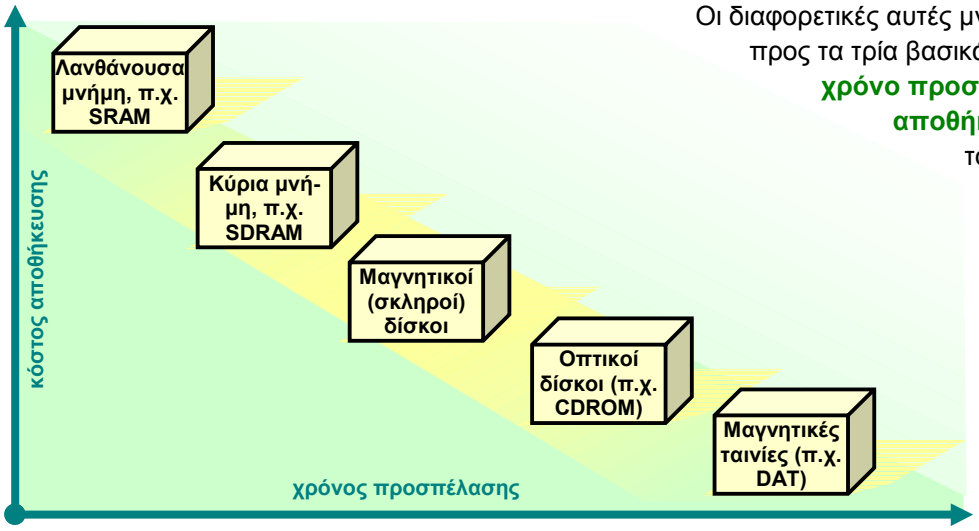
Όπως λοιπόν διαχειρίζεται ένας άνθρωπος τους τηλεφωνικούς αριθμούς, μπορεί και ο υπολογιστής να διαχειριστεί τα δεδομένα του: τα δεδομένα που χρησιμοποιούνται συχνά να βρίσκονται αποθηκευμένα σε μία μνήμη με μικρό μέγεθος και γρήγορη προσπέλαση, ενώ τα υπόλοιπα δεδομένα θα είναι αποθηκευμένα σε μνήμη με μεγαλύτερη χωρητικότητα για την οποία όμως απαιτείται πιο μεγάλος χρόνος προσπέλασης.

Στο εσωτερικό του επεξεργαστή αυτή η μέθοδος εφαρμόζεται με τη *λανθάνουσα μνήμη* (cache memory), την οποία θα δούμε αναλυτικά στο Μάθημα 4.4. Η μνήμη αυτή κρατά τα δεδομένα που χρησιμοποιούνται συχνά. Έτσι αυτά είναι άμεσα διαθέσιμα για χρήση και δεν είναι απαραίτητη μία εξωτερική επικοινωνία με την κύρια μνήμη.

Η ίδια μέθοδος εφαρμόζεται και στο σύστημα της *εικονικής μνήμης* (virtual memory), την οποία διαχειρίζεται το λειτουργικό σύστημα (θα παρουσιαστεί στο Κεφάλαιο 9). Εκεί η κύρια μνήμη του συστήματος παίζει το ρόλο της μικρής και γρήγορης μνήμης, ενώ η «αργή» και μεγαλύτερη σε μέγεθος μνήμη βρίσκεται σε κάποια περιφερειακή συσκευή, συνήθως στο σκληρό δίσκο.

Ιεραρχία μνήμης στους σύγχρονους υπολογιστές

Βλέπουμε ότι τα σύγχρονα υπολογιστικά συστήματα, για να καλύψουν διαφόρων επιπέδων ανάγκες σε ταχύτητα και χωρητικότητα μνήμης, επιστρατεύουν πολλαπλούς τύπους μνήμης.



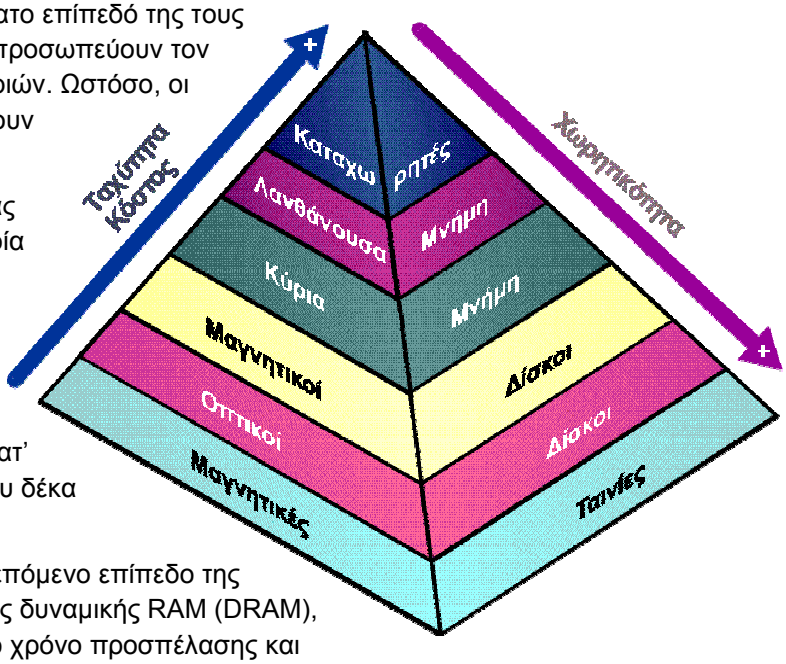
Οι διαφορετικές αυτές μνήμες διαφοροποιούνται ως προς τα τρία βασικά τους χαρακτηριστικά: το **χρόνο προσπέλασης**, το **κόστος αποθήκευσης** και τη **χωρητικότητά** τους. Μάλιστα, ο χρόνος προσπέλασης και η χωρητικότητα είναι «ανάλογα» μεταξύ τους, δηλαδή όταν το ένα αυξάνει, τότε αυξάνει και το άλλο, ενώ αυτά τα δύο μεγέθη είναι «αντιστρόφως ανάλογα» προς το κόστος αποθήκευσης.

Για να μπορεί να εκμεταλλευθεί ο υπολογιστής τα διαφορετικά χαρακτηριστικά των μνημών που διαθέτει, χωρίς το κόστος του να είναι παράλογο, οι μονάδες μνήμης μπορούν να θεωρηθούν ότι είναι οργανωμένες σε μια ιδεατή *ιεραρχία* (memory hierarchy). Στην «κορυφή» της ιεραρχίας, όπου το μέγεθος της μνήμης είναι πολύ μικρό, βρίσκονται οι πιο γρήγορες και ακριβές μνήμες. Στη «βάση» της ιεραρχίας το μέγεθος της μνήμης είναι μεγάλο και τοποθετούνται αργές και φθηνές μνήμες.

Η κύρια μνήμη ενός προσωπικού υπολογιστή είναι περίπου 1000 φορές πιο γρήγορη από ένα σκληρό δίσκο και γύρω στις 100 φορές πιο μικρή σε χωρητικότητα. Επίσης είναι περίπου 100 φορές πιο ακριβή ανά bit.

Με τον τρόπο αυτό ελαχιστοποιείται ο μέσος χρόνος προσπέλασης στη μνήμη και το κόστος αποθήκευσης ανά μονάδα πληροφορίας, ενώ μεγιστοποιείται ο συνολικός χώρος αποθήκευσης του συστήματος.

Μια τυπική ιεραρχία μνήμης έχει στο ανώτατο επίπεδό της τους **καταχωρητές** της ΚΜΕ, επειδή αυτοί αντιπροσωπεύουν τον αμεσότερο τρόπο αποθήκευσης πληροφοριών. Ωστόσο, οι καταχωρητές δεν μπορούν να αποθηκεύσουν αξιόλογο όγκο δεδομένων.



Στο αμέσως επόμενο επίπεδο της ιεραρχίας τοποθετείται η **λανθάνουσα μνήμη**, η οποία συνήθως υλοποιείται με ψηφίδες στατικής RAM (SRAM). Στη λανθάνουσα μνήμη αντιγράφονται τμήματα της κύριας μνήμης, με την προσδοκία ότι οι επόμενες αναφορές προς την κύρια μνήμη θα μπορέσουν να εξυπηρετηθούν κατ' ευθείαν.Η λανθάνουσα μνήμη είναι περίπου δέκα φορές ταχύτερη από την κύρια μνήμη.

Η **κύρια μνήμη** καταλαμβάνει το αμέσως επόμενο επίπεδο της ιεραρχίας. Υλοποιείται συνήθως με ψηφίδες δυναμικής RAM (DRAM), οι οποίες χαρακτηρίζονται από μεγαλύτερο χρόνο προσπέλασης και μικρότερο κόστος ανά μονάδα αποθηκευμένης πληροφορίας σε σχέση με τα κυκλώματα τύπου SRAM. Η κύρια μνήμη χρησιμοποιείται, όταν μία λέξη δεν βρεθεί στη λανθάνουσα μνήμη. Τότε, εκτός από την ανάγνωση ή εγγραφή αυτής της λέξης, ενεργοποιείται και ο μηχανισμός για την αντιγραφή των γειτονικών της θέσεων από την κύρια προς τη λανθάνουσα μνήμη. Αυτό γίνεται γιατί η τοπικότητα της αναφοράς προβλέπει ότι είναι πιθανόν οι επόμενες προσπελάσεις στη μνήμη να γίνουν σε λέξεις γειτονικές με αυτή που τώρα προσπελάστηκε.

Στα τελευταία επίπεδα της ιεραρχίας τοποθετούνται οι μονάδες των **μαγνητικών και οπτικών δίσκων**, και οι μονάδες των **μαγνητικών ταινιών**. Αυτές οι μονάδες εξασφαλίζουν τη μόνιμη και ασφαλή αποθήκευση μεγάλου όγκου δεδομένων. Η ταχύτητα προσπέλασης σε αυτές είναι σχετικά χαμηλή, λόγω των ηλεκτρομηχανικών διαδικασιών προσπέλασης στα δεδομένα. Οι μαγνητικές ταινίες είναι ιδεώδεις για την αποθήκευση *εφεδρικών αντιγράφων* (backup copies) των δεδομένων. Οι μαγνητικοί δίσκοι προσφέρονται και για την υλοποίηση του μηχανισμού της εικονικής μνήμης.

Ανακεφαλαίωση

Η τοπικότητα της αναφοράς βοηθά στην οργάνωση του συστήματος μνήμης με τρόπο που επιταχύνει την εκτέλεση των προγραμμάτων. Τα υπολογιστικά συστήματα διαθέτουν πολλούς τύπους μνήμης, που διαφέρουν ως προς τρία χαρακτηριστικά: τη χωρητικότητα, το κόστος και την ταχύτητα προσπέλασης. Οι διάφοροι τύποι μνήμης σχηματίζουν μια ιεραρχία, στην κορυφή της οποίας βρίσκονται οι καταχωρητές της ΚΜΕ και στη βάση της οι αργές και φθηνές μνήμες όπως οι μαγνητικές ταινίες.

Γλωσσάριο όρων

Εικονική Μνήμη	*Virtual Memory*
Εφεδρικό Αντίγραφο	*Backup Copy*
Ιεραρχία Μνήμης	*Memory Hierarchy*
Λανθάνουσα Μνήμη	*Cache Memory*
Τοπικότητα Αναφοράς	*Locality Of Reference*

Ερωτήσεις

? Τι είναι η τοπικότητα της αναφοράς στα προγράμματα;

? Πώς είναι οργανωμένη η ιεραρχία της μνήμης;

? Ποιο είδος μνήμης βρίσκεται στην κορυφή της ιεραρχίας της μνήμης;

? Ποια μεγέθη αυξάνονται όσο ανεβαίνουμε σε ανώτερα επίπεδα της ιεραρχίας της μνήμης;

? Ποια μεγέθη μειώνονται όσο ανεβαίνουμε σε ανώτερα επίπεδα της ιεραρχίας της μνήμης;

? Για ποιες λειτουργίες είναι κατάλληλες οι μνήμες στο κατώτατο επίπεδο της ιεραρχίας της μνήμης;

Μάθημα 4.3 Οργάνωση και Σχεδίαση Μνήμης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τους τρόπους σχεδίασης ενός συστήματος κύριας μνήμης.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να καταρτίζεις το Χάρτη Διευθύνσεων για ένα σύστημα μνήμης*
- *Να περιγράφεις την επικοινωνία του επεξεργαστή με την κάθε ψηφίδα*
- *Να εξηγείς τον τρόπο λειτουργίας ενός συστήματος με διαφύλλωση μνήμης*

Σχεδίαση συστήματος κύριας μνήμης

Η κύρια μνήμη σε ένα υπολογιστικό σύστημα αποτελείται από τη *μνήμη μόνο ανάγνωσης* (ROM) και από τη *μνήμη ανάγνωσης - εγγραφής* (RAM)[1].

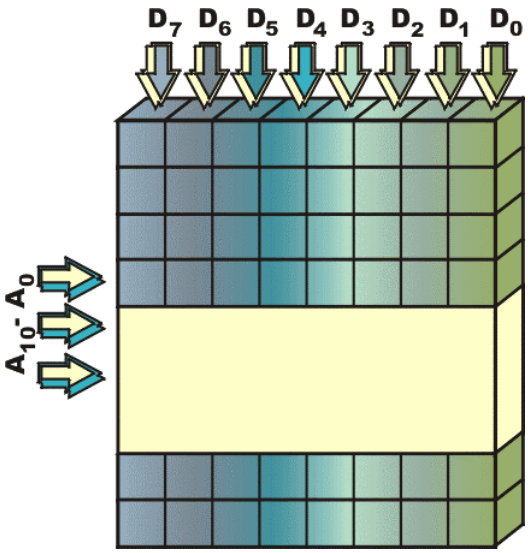
Η κύρια μνήμη είναι άμεσα προσπελάσιμη από τον επεξεργαστή, χρησιμοποιώντας τα σήματα ανάγνωσης (RD), εγγραφής (WR), Διευθύνσεων (A_{k-1} - A_0) και Δεδομένων (D_{n-1} - D_0), όπως είδαμε στο Μάθημα 4.1, σύμφωνα με τον κύκλο ανάγνωσης και τον κύκλο εγγραφής της μνήμης.

Για τη δημιουργία της κύριας μνήμης συνήθως χρησιμοποιούνται ψηφίδες ολοκληρωμένων κυκλωμάτων μνημών τύπου ROM και RAM. Οι ψηφίδες αυτές χαρακτηρίζονται από τις *διαστάσεις* (dimensions) τους, δηλαδή από το πλήθος των λέξεων που περιέχουν και το μήκος σε bits της κάθε λέξης.

> Μια ψηφίδα διαστάσεων 1024×8 περιέχει 1024 λέξεις των 8 bits η κάθε μία.

Με την κατάλληλη διασύνδεση τέτοιων ψηφίδων είναι δυνατή η σχεδίαση μονάδων κύριας μνήμης οποιωνδήποτε διαστάσεων (πλήθος και μέγεθος λέξεων), για την εξυπηρέτηση των συγκεκριμένων απαιτήσεων ενός υπολογιστικού συστήματος.

> Ένα σύστημα στηρίζεται σε επεξεργαστή ο οποίος διαθέτει 8 γραμμές δεδομένων (D_7 – D_0) και 11 γραμμές διευθύνσεων (A_{10} – A_0). Ο χώρος μνήμης του υπολογιστή αυτού λοιπόν θα αποτελείται από 2^{11} = 2048 λέξεις των 8 bits.



[1] Ο όρος RAM ο οποίος σημαίνει Μνήμη Τυχαίας Προσπέλασης, έχει επικρατήσει ιστορικά, χωρίς να περιγράφει επακριβώς τη μνήμη στην οποία αναφερόμαστε

Θέλουμε να σχεδιάσουμε μια μονάδα κύριας μνήμης που να διαθέτει μνήμη τύπου RAM για τις διευθύνσεις από 0 έως 1023 και μνήμη τύπου ROM για τις διευθύνσεις από 1024 έως 2047. Διαθέτουμε ψηφίδες RAM διαστάσεων 256x8 bits και ROM διαστάσεων 512x4 bits.

Για τον σκοπό αυτό καταρτίζουμε τον *Χάρτη Διευθύνσεων* (Address Map), όπου καταγράφονται όλες οι αναγκαίες ψηφίδες μνήμης, μαζί με τους αντίστοιχες περιοχές του χώρου διευθύνσεων του επεξεργαστή.

Επειδή οι μνήμες ROM περιέχουν λέξεις των 4 bits, για κάθε λέξη της μνήμης που αντιστοιχεί στη ROM (8 bits) θα χρησιμοποιήσουμε 2 λέξεις των 4 bits με την ίδια διεύθυνση από δύο διαφορετικές ψηφίδες ROM.

Οι διευθύνσεις που καταλαμβάνονται από μνήμη RAM, οι 0-1023, απαιτούν 1024/256 = 4 ψηφίδες, τις RAM_0, RAM_1, RAM_2 και RAM_3. Οι διευθύνσεις που καταλαμβάνονται από μνήμη ROM, οι 1024-2047 απαιτούν 1024/512 = 2 ζεύγη ψηφίδων ROM, τα (ROM_0, ROM_1) και (ROM_2, ROM_3).

Ο χάρτης διευθύνσεων θα είναι λοιπόν:

Ψηφίδα		Διευθύνσεις												
		A_{10}	A_9	A_8	A_7	A_6	A_5	A_4	A_3	A_2	A_1	A_0		
RAM_0	από	0	0	0	0	0	0	0	0	0	0	0	=	$0_{(10)}$
	έως	0	0	0	1	1	1	1	1	1	1	1	=	$255_{(10)}$
RAM_1	από	0	0	1	0	0	0	0	0	0	0	0	=	$256_{(10)}$
	έως	0	0	1	1	1	1	1	1	1	1	1	=	$511_{(10)}$
RAM_2	από	0	1	0	0	0	0	0	0	0	0	0	=	$512_{(10)}$
	έως	0	1	0	1	1	1	1	1	1	1	1	=	$767_{(10)}$
RAM_3	από	0	1	1	0	0	0	0	0	0	0	0	=	$768_{(10)}$
	έως	0	1	1	1	1	1	1	1	1	1	1	=	$1023_{(10)}$
ROM_0 και ROM_1	από	1	0	0	0	0	0	0	0	0	0	0	=	$1024_{(10)}$
	έως	1	0	1	1	1	1	1	1	1	1	1	=	$1535_{(10)}$
ROM_2 και ROM_3	από	1	1	0	0	0	0	0	0	0	0	0	=	$1536_{(10)}$
	έως	1	1	1	1	1	1	1	1	1	1	1	=	$2047_{(10)}$

Από το χάρτη διευθύνσεων βλέπουμε για κάθε διεύθυνση μνήμης την ψηφίδα στην οποία αντιστοιχεί. Για να βρούμε την εσωτερική διεύθυνση μέσα στην ψηφίδα που πρέπει να προσπελαστεί, αφαιρούμε την αρχική διεύθυνση για την ψηφίδα.

Η διεύθυνση 777 βρίσκεται μέσα στο διάστημα 768-1023, άρα βρίσκεται στην ψηφίδα RAM_3. Η εσωτερική της διεύθυνση μέσα στην ψηφίδα είναι η 777-768 = 9.

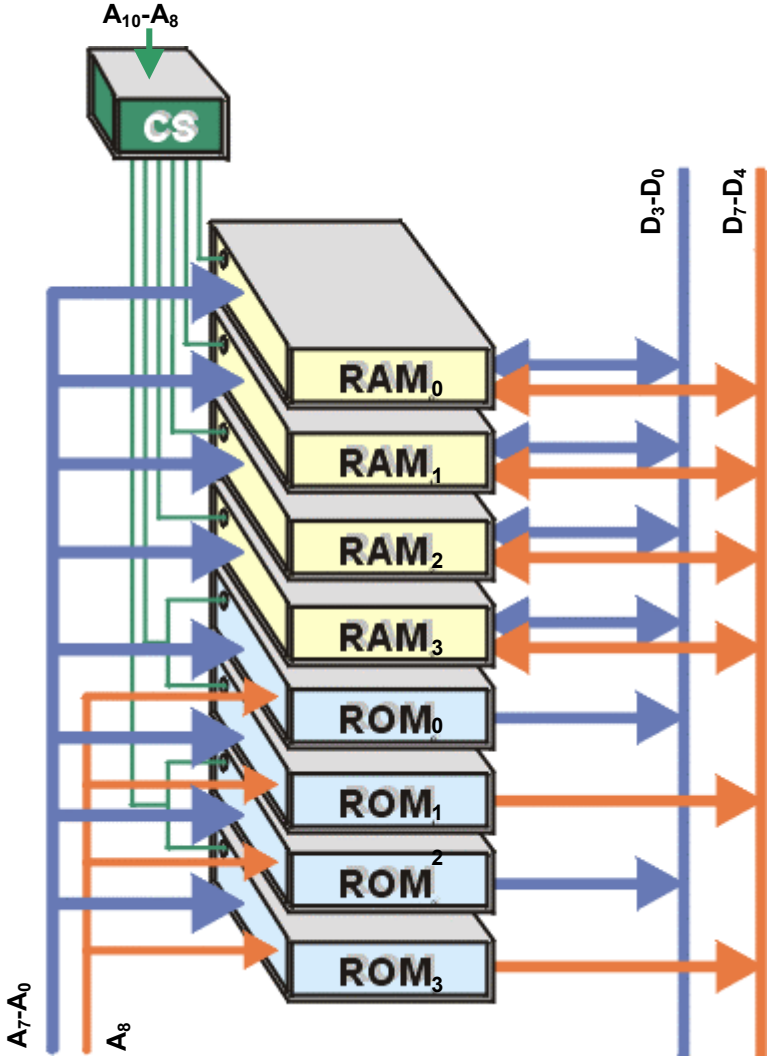
Η διεύθυνση 1548 καλύπτεται από τις ψηφίδες ROM_2 και ROM_3, από τις εσωτερικές θέσεις 1548-1536=12 κάθε ψηφίδας. Κάθε ψηφίδα παρέχει 4 από τα 8 bit της λέξης.

Προκειμένου να γίνεται επιλεκτική ενεργοποίηση των ψηφίδων της μνήμης, ανάλογα με την τιμή της διεύθυνσης, κάθε ψηφίδα έχει μια επιπλέον γραμμή εισόδου, την *επιλογή ψηφίδας* (chip select, CS).

Όταν η είσοδος CS είναι ανενεργή (δηλαδή έχει την τιμή 0), τότε η ψηφίδα μένει αδρανής και δεν ανταποκρίνεται σε κανένα από τα άλλα σήματα (RD ή WR). Μόνον όταν η είσοδος CS είναι ενεργή (έχει την τιμή 1) η ψηφίδα της μνήμης ανταποκρίνεται, σύμφωνα με τον κύκλο ανάγνωσης μνήμης (όταν RD=1) ή με τον κύκλο εγγραφής μνήμης (όταν WR=1), όπως είδαμε στο Μάθημα 4.1.

Για τη σχεδίαση του συστήματος κύριας μνήμης του παραδείγματός μας, πρέπει αρχικά να συνδέσουμε τις γραμμές δεδομένων όλων των ψηφίδων (D_7 – D_0) με τις αντίστοιχες γραμμές τού επεξεργαστή. Οι ψηφίδες ROM συνδέονται ανά ζεύγη, έτσι ώστε να καλύπτεται όλο το μήκος λέξης του υπολογιστή.

Οι γραμμές διευθύνσεως των ψηφίδων μνήμης συνδέονται επίσης με τις αντίστοιχες γραμμές διευθύνσεως του υπολογιστή. Οι ψηφίδες RAM που έχουν 256 λέξεις, διαθέτουν 8 γραμμές διευθύνσεως, οι οποίες συνδέονται με τις γραμμές A_0 έως A_7 του υπολογιστή. Αντίστοιχα, οι ψηφίδες ROM που έχουν 512 λέξεις, διαθέτουν 9 γραμμές διευθύνσεως, οι οποίες συνδέονται με τις γραμμές A_0 έως A_8 του υπολογιστή. Επίσης, όλες οι ψηφίδες RAM συνδέονται με τα σήματα RD και WR του υπολογιστή, για τον προσδιορισμό της λειτουργίας που πρέπει να εκτελεστεί κάθε φορά.



Επειδή κάθε διεύθυνση που δημιουργείται από τον επεξεργαστή (και μεταφέρεται μέσω των γραμμών διευθύνσεως) αναφέρεται σε συγκεκριμένη ψηφίδα (ή ζεύγος ψηφίδων), σύμφωνα με τον Χάρτη Διευθύνσεων, είναι απαραίτητος ένας επιπλέον μηχανισμός (CS), ο οποίος να επιλέγει και να ενεργοποιεί κάθε φορά την κατάλληλη ψηφίδα, ώστε αυτή μόνο να μεταφέρει δεδομένα πάνω στις γραμμές δεδομένων του επεξεργαστή.

Ο μηχανισμός CS θα επιλέγει την ψηφίδα RAM_3 μόνο όταν A_{10}=0, A_9=1 και A_8=1.

A_{10}	A_9	A_8	Ψηφίδα
0	0	0	RAM_0
0	0	1	RAM_1
0	1	0	RAM_2
0	1	1	RAM_3
1	0	0 ή 1	ROM_0 και ROM_1
1	1	0 ή 1	ROM_2 και ROM_3

Μπορούμε να δούμε από τον Χάρτη Διευθύνσεων ότι οι τιμές των ψηφίων A_{10}, A_9 και A_8 μπορούν να χρησιμοποιηθούν για την ενεργοποίηση των ψηφίδων μνήμης, σύμφωνα με τον πίνακα.

Διαφύλλωση μνήμης

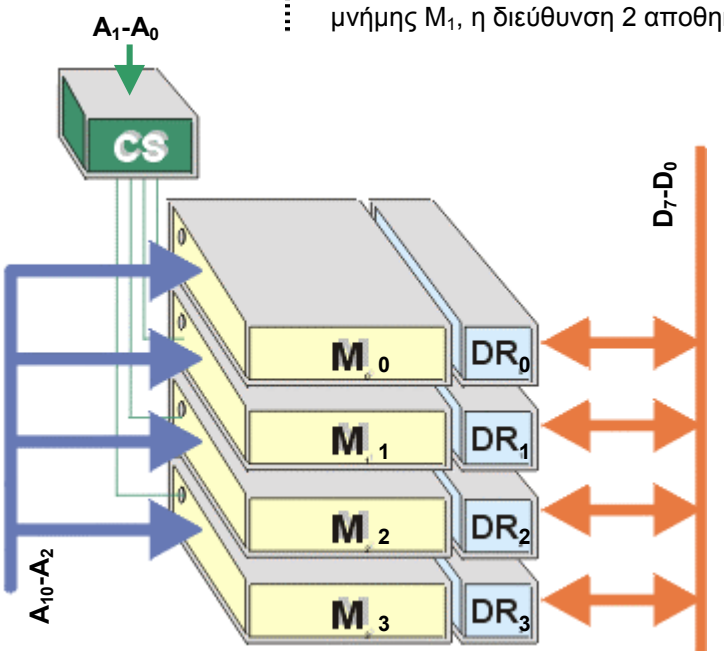
Στο παράδειγμα της προηγούμενης παραγράφου, είδαμε ότι οι γειτονικές διευθύνσεις μνήμης αποθηκεύονται στην ίδια ψηφίδα. Μπορούμε όμως να αντιστοιχίσουμε γειτονικές διευθύνσεις μνήμης σε διαφορετικές ψηφίδες με τέτοιο τρόπο, ώστε, υπό ορισμένες προϋποθέσεις, να μειωθεί ο μέσος χρόνος που απαιτείται για μια αναφορά στη μνήμη.

Η τεχνική κατά την οποία η απεικόνιση γειτονικών θέσεων του χώρου μνήμης γίνεται σε διαφορετικές ψηφίδες, αναφέρεται ως *διαφύλλωση μνήμης* (memory interleaving).

Θέλουμε να συνδυάσουμε 4 ψηφίδες μνήμης (M_0, M_1, M_2 και M_3), διαστάσεων 512×8 bits η κάθε μια, για τη δημιουργία ενός συστήματος μνήμης 2K (=2048) λέξεων.

Κάθε μια από τις ψηφίδες M τοποθετεί σε ένα εσωτερικό της καταχωρητή, τον *καταχωρητή δεδομένων εξόδου* (Data Register, DR) το περιεχόμενο της λέξης που υποδεικνύεται από την τιμή των γραμμών διευθύνσεών της. Το περιεχόμενο του καταχωρητή DR μεταφέρεται στις γραμμές δεδομένων του συστήματος μόνον όταν ενεργοποιηθεί η αντίστοιχη είσοδος επιλογής CS.

Με τη συνδεσμολογία του σχήματος, η διεύθυνση 0 αποθηκεύεται στη λέξη 0 της μνήμης M_0, η διεύθυνση 1 αποθηκεύεται στη λέξη 0 της μνήμης M_1, η διεύθυνση 2 αποθηκεύεται στη λέξη 0 της μνήμης M_2 και η διεύθυνση 3 αποθηκεύεται στη λέξη 0 της μνήμης M_3. Ανάλογα, προκύπτει ότι διεύθυνση 4 αποθηκεύεται στη θέση 1 της μνήμης M_0 κ.ο.κ.



Όταν γίνεται ανάγνωση μιας λέξης από τη μνήμη, ταυτόχρονα διαβάζονται και οι άλλες τρεις γειτονικές της λέξεις, από τις άλλες τρεις ψηφίδες.

Για παράδειγμα, όταν ζητηθεί η ανάγνωση από τη διεύθυνση 5 (**00000000101**), αυτή η λέξη θα δοθεί από τη μνήμη M_1, μέσω του DR_1.Ταυτόχρονα όμως, θα διαβαστούν και οι γειτονικές της διευθύνσεις 4, 6 και 7. Τα περιεχόμενα των διευθύνσεων αυτών θα παραμείνουν στους καταχωρητές DR_0, DR_2 και DR_3,

αντίστοιχα. Έτσι, εάν τύχει η επόμενη αναφορά μνήμης να είναι για ανάγνωση από κάποια από τις διευθύνσεις 4, 5, 6 ή 7, τότε αυτή η λέξη διαβάζεται απευθείας από τον αντίστοιχο καταχωρητή DR, χωρίς την καθυστέρηση που απαιτείται για την μεταφορά της λέξης από το εσωτερικό της μνήμης στον καταχωρητή DR. Δεδομένου ότι οι περισσότερες αναφορές στη μνήμη ακολουθούν την ιδιότητα της τοπικότητας της αναφοράς, και επειδή η πλειοψηφία των αναφορών στη μνήμη γίνεται για ανάγνωση δεδομένων, παρατηρείται σημαντική βελτίωση του μέσου χρόνου προσπέλασης της μνήμης με τη χρήση της διαφύλλωσης.

Στην περίπτωση του παραδείγματός μας έχουμε *διαφύλλωση 4 δρόμων* (4-way interleaving), επειδή γίνεται ταυτόχρονη ανάκτηση 4 γειτονικών θέσεων. Το πλήθος των δρόμων είναι κάποια δύναμη του δύο· όσο μεγαλύτερο το πλήθος των δρόμων, τόσο μεγαλύτερη η πιθανότητα για ταχεία ανάγνωση μιας λέξεως από τη μνήμη.

Γενικά λοιπόν, ένα σύστημα μνήμης με *διαφύλλωση κ-δρόμων* (k-way interleaving), όπου $\kappa=2^m$, αποτελείται από κ ψηφίδες μνήμης. Η επιλογή της ψηφίδας γίνεται από τα m δεξιότερα ψηφία της διεύθυνσης, και η εσωτερική διεύθυνση της ψηφίδας δίνεται από τα υπόλοιπα bits της διεύθυνσης μνήμης. Με τον τρόπο αυτό, οι κ συνεχόμενες θέσεις μνήμης αποθηκεύονται σε κ διαφορετικές ψηφίδες. Όταν γίνεται ανάγνωση για οποιαδήποτε λέξη της μνήμης, τότε μεταφέρονται στους εσωτερικούς καταχωρητές (DR) των ψηφίδων της μνήμης οι υπόλοιπες κ-1 γειτονικές λέξεις. Έτσι, αν στη συνέχεια διαβαστεί μια από αυτές, κάτι που γίνεται συχνά, θα είναι αμέσως διαθέσιμη.

Ανακεφαλαίωση

Για να σχεδιάσουμε ένα σύστημα κύριας μνήμης για ένα συγκεκριμένο επεξεργαστή από δεδομένους τύπους ψηφίδων, πρώτα καταρτίζουμε το Χάρτη Διευθύνσεων. Από το χάρτη αυτό βρίσκουμε τον τρόπο σύνδεσης των ψηφίδων με τις γραμμές διευθύνσεων και δεδομένων της ΚΜΕ. Αν θέλουμε να επιταχύνουμε τις προσπελάσεις σε γειτονικές θέσεις μνήμης, χρησιμοποιούμε τη μέθοδο της διαφύλλωσης της μνήμης, που κατανέμει γειτονικές θέσεις μνήμης σε διαφορετικές ψηφίδες, ώστε να είναι όλες διαθέσιμες με μια ανάγνωση.

Γλωσσάριο όρων

Διαστάσεις	*Dimensions*
Διαφύλλωση κ-Δρόμων	*k-Way Interleaving*
Διαφύλλωση Μνήμης	*Memory Interleaving*
Επιλογή Ψηφίδας	*Chip Select - CS*
Καταχωρητής Δεδομένων Εξόδου	*Data Register - DR*
Μνήμη Μόνο Ανάγνωσης	*Read Only Memory - ROM*
Μνήμη Ανάγνωσης-Εγγραφής	*RAM*
Χάρτης Διευθύνσεων	*Address Map*

Ερωτήσεις

? Τι παριστάνουν οι διαστάσεις μιας ψηφίδας μνήμης;

? Πώς καταρτίζουμε το χάρτη διευθύνσεων για ένα σύστημα μνήμης;

? Ποια είναι η λειτουργία της επιλογής ψηφίδας;

? Τι είναι η διαφύλλωση μνήμης;

? Τι επιτυγχάνουμε με τη διαφύλλωση μνήμης;

Μάθημα 4.4 Λανθάνουσα Μνήμη

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει την οργάνωση της λανθάνουσας μνήμης και τον τρόπο που επιταχύνει την εκτέλεση των προγραμμάτων.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις την οργάνωση της λανθάνουσας μνήμης*
- *Να καθορίζεις πότε αντιγράφονται δεδομένα από την κύρια στη λανθάνουσα μνήμη*
- *Να υπολογίζεις πόσο επιταχύνεται η ταχύτητα εκτέλεσης ενός προγράμματος από την ύπαρξη της λανθάνουσας μνήμης*

Αρχές λειτουργίας της λανθάνουσας μνήμης

Η *λανθάνουσα μνήμη* (cache memory) ανήκει στα ανώτερα στρώματα της ιεραρχίας της μνήμης, επειδή χαρακτηρίζεται από τη **μεγάλη ταχύτητα προσπέλασης** αλλά και το **υψηλό κόστος κατασκευής**.

Συνήθως υλοποιείται με την τεχνολογία των στατικών μνημών (SRAM), που διαθέτουν τα δύο αυτά χαρακτηριστικά. Επίσης οι στατικές μνήμες έχουν μικρή πυκνότητα αποθήκευσης, ιδιαίτερα σε σχέση με τις δυναμικές μνήμες, που σε συνδυασμό με το υψηλό κόστος τους δεν τις καθιστούν κατάλληλες για τη δημιουργία όλου του χώρου της κύριας μνήμης.

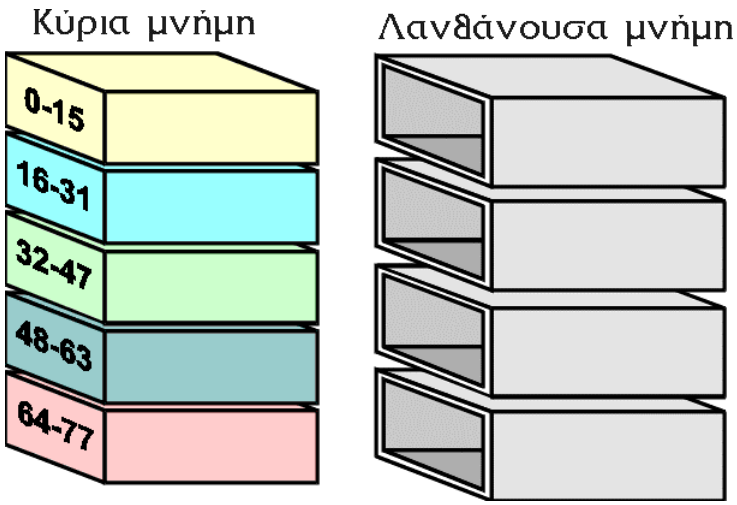
Αυτό συνεχίζει να ισχύει και σήμερα, παρά τη διαρκή μείωση των τιμών για όλους τύπους μνήμης, επειδή παράλληλα διογκώνονται συνεχώς και οι απαιτήσεις σε χώρο μνήμης όλων των σύγχρονων εφαρμογών και λειτουργικών συστημάτων. Έτσι, προτιμάται η χρήση και των δύο τεχνολογιών μνήμης, δυναμικής και στατικής, για την κύρια και λανθάνουσα μνήμη αντίστοιχα.

Όπως είδαμε και στο Μάθημα 4.2, η λειτουργία της λανθάνουσας μνήμης στηρίζεται στην αντιγραφή μέρους από τα περιεχόμενα της κύριας μνήμης. Αξιοποιώντας την τοπικότητα της αναφοράς των προγραμμάτων, οι περισσότερες αναφορές σε θέσεις μνήμης εξυπηρετούνται από τη λανθάνουσα μνήμη με πολύ μικρή καθυστέρηση.

Για τον σκοπό αυτό η λανθάνουσα μνήμη είναι οργανωμένη σε ενότητες λέξεων, που αναφέρονται και ως *γραμμές* (cache lines). Κάθε γραμμή της λανθάνουσας μνήμης έχει σταθερό μέγεθος (π.χ. 16 λέξεις) και περιλαμβάνει τα αντίγραφα λέξεων, που είναι γειτονικές στην κύρια μνήμη. Έτσι, μπορούμε να φανταστούμε ότι η κύρια μνήμη χωρίζεται σε αντίστοιχες νοητές ενότητες, μεγέθους όσο μια γραμμή της λανθάνουσας μνήμης. Ορισμένες από τις ενότητες αυτές μπορούν να αντιγραφούν στη λανθάνουσα μνήμη.

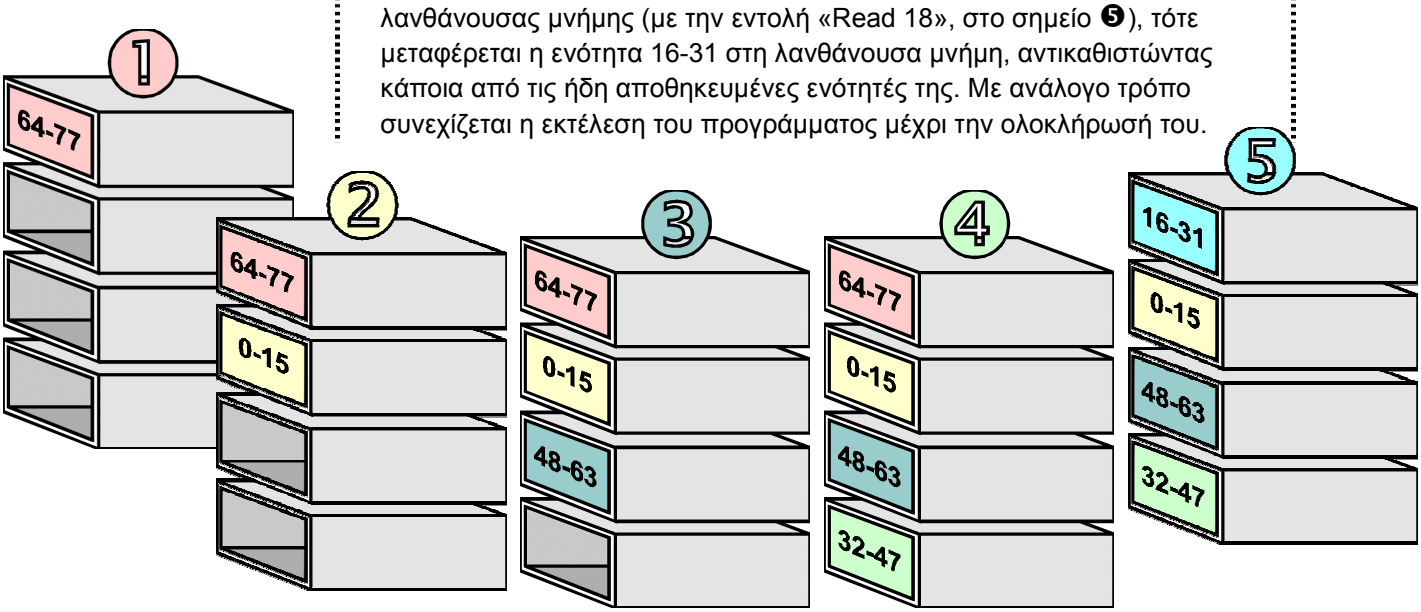
read 65 ①
read 66
read 8 ②
read 67
read 9
read 65
read 66
write 51 ③
read 7
read 8
read 33 ④
read 34
write 52
read 35
read 18 ⑤
read 36

Ας δούμε ένα σενάριο προσπελάσεων μνήμης από ένα υποθετικό πρόγραμμα. Η λανθάνουσα μνήμη αποτελείται από 64 λέξεις, χωρισμένες σε 4 γραμμές των 16 λέξεων η κάθε μια, και αρχικά είναι κενή.



Κάθε φορά που σημειώνεται *αποτυχία της λανθάνουσας μνήμης* (cache miss), δηλαδή αδυναμία να βρεθεί η ζητούμενη λέξη σε κάποια γραμμή της, μεταφέρεται ολόκληρη η αντίστοιχη ενότητα από την κύρια μνήμη στην λανθάνουσα μνήμη.

Αυτό συμβαίνει στα σημεία ❶,❷,❸ και ❹ του προγράμματος. Μετά από αυτά, όλες οι ενότητες της λανθάνουσας μνήμης είναι κατειλημμένες με περιεχόμενα της κύριας μνήμης. Όταν παρατηρηθεί η επόμενη αποτυχία λανθάνουσας μνήμης (με την εντολή «Read 18», στο σημείο ❺), τότε μεταφέρεται η ενότητα 16-31 στη λανθάνουσα μνήμη, αντικαθιστώντας κάποια από τις ήδη αποθηκευμένες ενότητές της. Με ανάλογο τρόπο συνεχίζεται η εκτέλεση του προγράμματος μέχρι την ολοκλήρωσή του.



Όλες οι αναγκαίες λειτουργίες για τη διαχείριση της λανθάνουσας μνήμης γίνονται εντελώς αυτόματα από το υλικό του επεξεργαστή. Οι λειτουργίες αυτές είναι:

- ο έλεγχος των περιεχομένων της λανθάνουσας μνήμης·
- οι μεταφορές των δεδομένων από την κύρια μνήμη·
- οι αντικαταστάσεις των δεδομένων όταν η λανθάνουσα μνήμη είναι πλήρης.

Έτσι δεν προκαλείται αξιόλογη καθυστέρηση στο πρόγραμμα που εκτελείται, γιατί κάτι τέτοιο θα εκμηδένιζε τα πλεονεκτήματα της λανθάνουσας μνήμης.

Χρήση της λανθάνουσας μνήμης

Όταν ξεκινά μια ανάγνωση από ή μια εγγραφή στη μνήμη, γίνονται τα ακόλουθα βήματα:

- Ελέγχεται άν η ζητούμενη λέξη είναι αποθηκευμένη στη λανθάνουσα μνήμη.
- Εάν αυτό ισχύει, τότε η αναφορά στη μνήμη ολοκληρώνεται ταχύτατα από τη λανθάνουσα μνήμη, χωρίς την ανάμιξη της κύριας μνήμης.
- Εάν η ζητούμενη διεύθυνση μνήμης δεν περιλαμβάνεται στα τρέχοντα περιεχόμενα της λανθάνουσας μνήμης, τότε η προσπέλαση (ανάγνωση ή εγγραφή) γίνεται στην κύρια μνήμη. Ταυτόχρονα αντιγράφονται από την κύρια μνήμη όλες οι λέξεις της ενότητας στην οποία ανήκει η ζητούμενη λέξη. Αυτό γίνεται επειδή, σύμφωνα με την αρχή της **τοπικότητας της αναφοράς**, οι επόμενες αναφορές στη μνήμη είναι πολύ πιθανό να γίνουν σε διευθύνσεις γειτονικές με την προηγούμενη.

Με τη διαδικασία αυτή, ένα μεγάλο ποσοστό από τις αναφορές στην μνήμη εξυπηρετείται ταχύτατα μέσω της λανθάνουσας μνήμης. Αυτό το ποσοστό, που συνήθως αναφέρεται και ως *λόγος επιτυχίας* λ (hit ratio), είναι δυνατόν να ξεπεράσει το 99%, οδηγώντας έτσι σε σημαντική μείωση του χρόνου εκτέλεσης των προγραμμάτων.

> Εάν ο χρόνος προσπέλασης στην κύρια μνήμη είναι 100 ns, ο χρόνος προσπέλασης στη λανθάνουσα μνήμη είναι 10 ns, και ο λόγος επιτυχίας είναι λ=95%, τότε ο μέσος χρόνος προσπέλασης t_μ στη μνήμη είναι:
>
> $$t_\mu = \frac{95}{100} \times 10ns + \frac{5}{100} \times (100ns + 10ns) = 9{,}5ns + 5{,}5ns = 15ns,$$
>
> επειδή το 95% των προσπελάσεων γίνεται στη λανθάνουσα μνήμη με ταχύτητα 10ns, ενώ για το 5% των προσπελάσεων γίνεται αναφορά και στη λανθάνουσα μνήμη και στην κύρια μνήμη με ταχύτητα 10ns + 100ns.

Στη γενική περίπτωση, εάν θεωρήσουμε ότι t_m είναι ο χρόνος προσπέλασης στην κύρια μνήμη, t_c ο χρόνος προσπέλασης στη λανθάνουσα μνήμη και λ είναι ο λόγος επιτυχίας, τότε ο μέσος χρόνος προσπέλασης στη μνήμη είναι:

$$t_\mu = \frac{\lambda}{100} \times t_c + (1 - \frac{\lambda}{100}) \times (t_c + t_m)$$

Παρατηρούμε ότι όσο ο λόγος επιτυχίας λ πλησιάζει το 100, η τιμή του μέσου χρόνου προσπέλασης στη μνήμη (t_μ) τείνει να εξισωθεί με τον χρόνο προσπέλασης της λανθάνουσας μνήμης (t_c).

Μέθοδοι εγγραφής στη λανθάνουσα μνήμη

Σε ένα σύστημα με λανθάνουσα μνήμη η λειτουργία της εγγραφής στη μνήμη μπορεί να αντιμετωπιστεί με διαφορετικούς τρόπους[1].

> Η εντολή «Write 52», στο προηγούμενο παράδειγμα, οδηγεί στην καταγραφή μιας νέας λέξης στη διεύθυνση μνήμης 52, η οποία τυχαίνει

[1] Σημειώνουμε ότι το πλήθος των εγγραφών στη μνήμη είναι πολύ μικρότερο από αυτό των αναγνώσεων.

να είναι αποθηκευμένη και στη λανθάνουσα μνήμη (ενότητα 48-63). Τόσο τα περιεχόμενα της κύριας όσο και της λανθάνουσας μνήμης πρέπει να ενημερωθούν.

Στην περίπτωση αυτή το σύστημα μπορεί να εφαρμόσει μια από τις ακόλουθες δύο τεχνικές:

α) ***Επανεγγραφή*** (write back): Στην περίπτωση αυτή, η νέα τιμή της θέσης μνήμης 52 καταγράφεται μόνο στη λανθάνουσα μνήμη. Όταν όμως αργότερα χρειαστεί να απομακρυνθεί η ενότητα 48-63 από τη λανθάνουσα μνήμη, θα πρέπει τα περιεχόμενά της να αντιγραφούν πάλι στις αντίστοιχες θέσεις της κύριας μνήμης. Η μέθοδος αυτή έχει ως κύριο πλεονέκτημα την πολύ γρήγορη εκτέλεση των εγγραφών στη μνήμη. Απαιτεί, ωστόσο, πρόσθετο έργο για την αντιγραφή από τη λανθάνουσα μνήμη στην κύρια μνήμη[2] των ενοτήτων που αντικαθίστανται.

β) ***Διεγγραφή*** (write through): Στην περίπτωση αυτή, η νέα τιμή της θέσης μνήμης 52 καταγράφεται στη λανθάνουσα μνήμη και ταυτόχρονα, ενώ συνεχίζεται η εκτέλεση του προγράμματος με επόμενες λειτουργίες, γίνεται καταγραφή της νέας τιμής και στη διεύθυνση 52 της κύριας μνήμης. Η μέθοδος αυτή εξασφαλίζει τη μόνιμη συμφωνία ανάμεσα στα περιεχόμενα της λανθάνουσας μνήμης και της κύριας μνήμης. Έτσι, όταν μια ενότητα αντικαθίσταται στη λανθάνουσα μνήμη, δε χρειάζεται να επανεγγραφεί στην κύρια μνήμη. Μπορεί όμως να προκαλέσει καθυστέρηση στην ολοκλήρωση των εγγραφών στη μνήμη.

Αντικατάσταση ενοτήτων στη λανθάνουσα μνήμη

Δεν είναι δύσκολο να δούμε ότι η επιλογή των ενοτήτων που θα μεταφερθούν στη λανθάνουσα μνήμη επηρεάζει σε μεγάλο βαθμό το λόγο επιτυχίας της λανθάνουσας μνήμης και συνεπώς την ταχύτητα εκτέλεσης των προγραμμάτων.

Όταν εκτελείται ένα πρόγραμμα, ενότητες από την κύρια μνήμη μεταφέρονται στη λανθάνουσα μνήμη έως ότου αυτή γεμίσει. Αφού συμβεί αυτό, όλες οι επόμενες αποτυχίες στη λανθάνουσα μνήμη αναγκαστικά οδηγούν στην αντικατάσταση μιας από τις υπάρχουσες ενότητες με αυτή στην οποία παρουσιάστηκε η αποτυχία.

Αν αντικατασταθεί (δηλαδή διαγραφεί) μια ενότητα και κατόπιν γίνει αναφορά σε κάποια λέξη της ενότητας αυτής, προκαλείται μια νέα αποτυχία λανθάνουσας μνήμης. Αυτό έχει προφανείς αρνητικές συνέπειες στην απόδοση του συστήματος μνήμης. Για το λόγο αυτό, επιδιώκεται η αντικατάσταση εκείνης της ενότητας για την οποία η πιθανότητα να γίνει αναφορά στο άμεσο μέλλον είναι ελάχιστη. Με αυτό το στόχο, και δεδομένου ότι μας είναι άγνωστο ποιες ακριβώς θα είναι οι επόμενες αναφορές μνήμης ενος προγράμματος, υπάρχουν οι ακόλουθες τακτικές αντικατάστασης:

- **Αντικατάσταση με βάση το χρόνο παραμονής** (First-In First-Out, FIFO), κατά την οποία αντικαθίσταται κάθε φορά εκείνη η ενότητα που είχε εισαχθεί παλαιότερα στη λανθάνουσα μνήμη.
- **Αντικατάσταση με βάση το χρόνο τελευταίας προσπέλασης** (Least Recently Used, LRU), κατά την οποία αντικαθίσταται κάθε φορά εκείνη η ενότητα στην οποία έχει να γίνει αναφορά για περισσότερο χρόνο.

---

[2] Αυτό μπορεί να αποφευχθεί μόνον αν, στην ενότητα που θα αντικατασταθεί, δεν έχει γίνει καμία εγγραφή για όσο χρόνο η ενότητα αυτή βρισκόταν στη λανθάνουσα μνήμη.

Και οι δύο αυτές μέθοδοι προσπαθούν να εκμεταλλευθούν την τοπικότητα αναφοράς των προγραμμάτων, προκειμένου να προβλέψουν την ενότητα με τη μικρότερη πιθανότητα αναφοράς. Από αυτήν την άποψη, η μέθοδος LRU είναι η πιο αποτελεσματική. Η υλοποίησή της, όμως, είναι πιο πολύπλοκη από ό,τι η FIFO, επειδή πρέπει να κρατά και να επεξεργάζεται στοιχεία για το χρόνο στον οποίο γίνεται προσπέλαση σε κάθε ενότητα της λανθάνουσας μνήμης.

Πολυεπίπεδη οργάνωση λανθάνουσας μνήμης

Με τη διαρκή πρόοδο στον τομέα των ολοκληρωμένων κυκλωμάτων έχει γίνει δυνατή η υλοποίηση ενος μέρους της λανθάνουσας μνήμης μέσα στο κύκλωμα του ίδιου του επεξεργαστή. Αυτή η μνήμη είναι πάρα πολύ γρήγορη, επειδή η προσπέλαση γίνεται στο εσωτερικό του ίδιου ολοκληρωμένου κυκλώματος, του επεξεργαστή.

Αυτή η «εσωτερική» μνήμη, όμως, είναι σχετικά περιορισμένη σε μέγεθος· έτσι, χρησιμοποιείται και άλλη, εξωτερική μνήμη, ως λανθάνουσα μνήμη, δημιουργώντας έτσι δύο επίπεδα λανθάνουσας μνήμης:

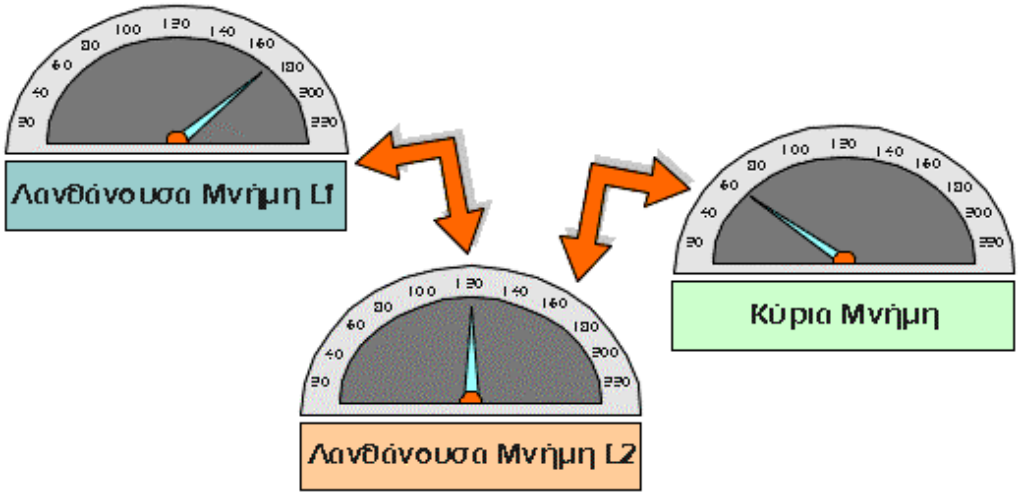
L1 Το *πρώτο επίπεδο λανθάνουσας μνήμης* (Level 1 cache ή L1) είναι πάρα πολύ ταχύ επειδή ανήκει στον επεξεργαστή και λειτουργεί με την ίδια ταχύτητα με αυτόν.

L2 Το *δεύτερο επίπεδο λανθάνουσας μνήμης* (Level 2 cache ή L2) έχει συνήθως περισσότερο χώρο αποθήκευσης και σχετικά μεγαλύτερο χρόνο προσπέλασης.

Το σύστημα διαχείρισης της λανθάνουσας μνήμης προσπαθεί να επιτύχει τη μέγιστη αξιοποίηση της μνήμης L1, ώστε να ελαχιστοποιείται ο χρόνος εκτέλεσης των προγραμμάτων.

Στην περίπτωση αυτή, η προσπέλαση της μνήμης γίνεται σε τρία βήματα:

- Αν η λέξη που προσπελάζεται βρίσκεται στη μνήμη L1, τότε χρησιμοποιείται απευθείας από εκεί.
- Αν δε βρεθεί στη μνήμη L1, τότε ελέγχεται μήπως βρίσκεται στη μνήμη L2. Αν ναι, χρησιμοποιείται και αντιγράφεται στη μνήμη L1.
- Αν δε βρεθεί ούτε στη μνήμη L2, τότε η λέξη προσπελάζεται στο χώρο της κύριας μνήμης και αντιγράφεται συγχρόνως στη μνήμη L2.



Ανακεφαλαίωση

Η λανθάνουσα μνήμη είναι μια γρήγορη μνήμη που κρατά τα πιο πρόσφατα χρησιμοποιημένα περιεχόμενα της κύριας μνήμης. Όταν γίνεται μία αναφορά στη μνήμη, καταβάλλεται προσπάθεια να εξυπηρετηθεί αυτή από τη λανθάνουσα μνήμη, και αν αυτό δε γίνει, τότε οι αντίστοιχες λέξεις αντιγράφονται από τη λανθάνουσα στην κύρια μνήμη. Το ποσοστό των αναφορών στη μνήμη που εξυπηρετούνται από τη λανθάνουσα μνήμη ονομάζεται ποσοστό επιτυχίας.

Γλωσσάριο όρων

Αντικατάσταση με Βάση το Χρόνο Παραμονής	*First-In First-Out - FIFO*
Αντικατάσταση με Βάση το Χρόνο Τελευταίας Προσπέλασης	*Least Recently Used - LRU*
Αποτυχία της Λανθάνουσας Μνήμης	*Cache Miss*
Γραμμή	*Cacheline*
Δεύτερο Επίπεδο Λανθάνουσας Μνήμης	*Level 2 Cache - L2*
Διεγγραφή	*Write Through*
Επανεγγραφή	*Write Back*
Λανθάνουσα Μνήμη	*Cache Memory*
Λόγος Επιτυχίας	*Hit Ratio*
Πρώτο Επίπεδο Λανθάνουσας Μνήμης	*Level 1 Cache - L1*

Ερωτήσεις

? Ποια είναι τα κύρια χαρακτηριστικά της λανθάνουσας μνήμης;

? Με τι τύπους κυκλωμάτων συνήθως υλοποιείται η λανθάνουσα μνήμη;

? Πώς ονομάζονται τα τμήματα στα οποία χωρίζεται η λανθάνουσα μνήμη;

? Τι είναι ο λόγος επιτυχίας λ;

? Ποιοι είναι οι τρόποι εγγραφής στη λανθάνουσα μνήμη;

? Ποιες είναι οι μέθοδοι αντικατάστασης γραμμών στη λανθάνουσα μνήμη;

? Τι είναι τα επίπεδα λανθάνουσας μνήμης L1 και L2;

Τι μάθαμε σε αυτό το κεφάλαιο

- Η μνήμη του υπολογιστή συνήθως οργανώνεται σε ομάδες δυαδικών ψηφίων, τις λέξεις.
- Οι διάφοροι τύποι μνήμης του υπολογιστή σχηματίζουν μία ιεραρχία, στην κορυφή της οποίας βρίσκονται οι καταχωρητές της ΚΜΕ και στη βάση της οι αργές και φθηνές μνήμες.
- Η κύρια μνήμη ενός υπολογιστή μπορεί να καταρτιστεί από διάφορες ψηφίδες, διασυνδέοντάς τις με τον επεξεργαστή όπως υπαγορεύει ο χάρτης διευθύνσεων.
- Με τη διαφύλλωση μνήμης αποθηκεύουμε γειτονικές διευθύνσεις μνήμης σε διαφορετικές ψηφίδες μνήμης.
- Η λανθάνουσα μνήμη είναι μία γρήγορη μνήμη που κρατά τα πιο πρόσφατα χρησιμοποιημένα περιεχόμενα της κύριας μνήμης για να επιταχύνει την εκτέλεση των προγραμμάτων.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Π. Τσανάκα, Γ. Φραγκάκη, Αρχιτεκτονική Υπολογιστών, Συμμετρία, 1987.
- Patterson D., Hennesy J., Computer Organization and Design: The Hardware/Software Interface, Morgan Kaufmann, 2η έκδοση, 1998.
- Hayes J., Computer Architecture and Organization, McGraw-Hill, 1988.

Κεφάλαιο

Περιφερειακές Μονάδες

Σκοπός του κεφαλαίου αυτού είναι να περιγράψει τον τρόπο διασύνδεσης και λειτουργίας καθώς και τις τεχνολογίες των βασικότερων περιφερειακών μονάδων που υποστηρίζουν τη λειτουργία του υπολογιστή.

Όταν ολοκληρώσεις το κεφάλαιο αυτό:

- Θα περιγράφεις την λειτουργία, την οργάνωση και τις κατηγορίες διαδρόμων στους σύγχρονους υπολογιστές.
- Θα εξηγείς τις σύγχρονες τεχνολογίες εκτύπωσης και τις διάφορες κατηγορίες εκτυπωτών που χρησιμοποιούνται από τους υπολογιστές.
- Θα κατανοήσεις τις πιο αντιπροσωπευτικές τεχνολογίες που χρησιμοποιούνται στις οθόνες των υπολογιστών
- Θα περιγράφεις τον τρόπο λειτουργίας, τις κατηγορίες και τα χαρακτηριστικά των οπτικών και μαγνητικών μέσων αποθήκευσης στους σύγχρονους υπολογιστές.
- Θα εξηγείς την λειτουργία και τις τεχνολογίες των πιο διαδεδομένων συσκευών διασύνδεσης

Μαθήματα

5.1 Διάδρομοι Υπολογιστικών Συστημάτων

5.2 Τεχνολογίες Εκτυπωτών

5.3 Τεχνολογία Απεικονιστικών Μονάδων

5.4 Μαγνητικά Μέσα Αποθήκευσης

5.5 Οπτικά Μέσα Αποθήκευσης

5.6 Συσκευές Τηλεπικοινωνιών & Δικτύωσης

Μάθημα 5.1 Διάδρομοι Υπολογιστικών Συστημάτων

Σκοπός του μαθήματος

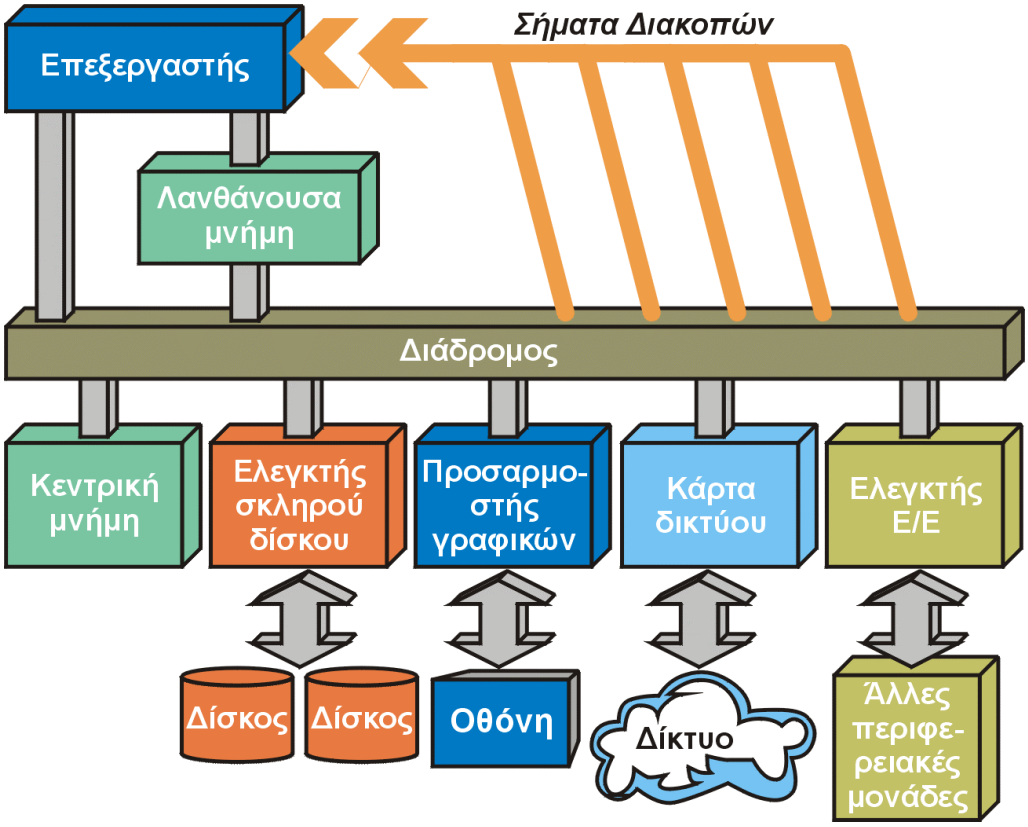
Σκοπός του μαθήματος είναι να περιγράψει τις λειτουργία και τις κατηγορίες διαδρόμων στους σύγχρονους υπολογιστές και ιδιαίτερα την οργάνωση των επιμέρους διαδρόμων στους προσωπικούς υπολογιστές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να ορίζεις τι είναι διάδρομος, και ποια τα χαρακτηριστικά του*
- *Να διακρίνεις τις κατηγορίες διαδρόμων*
- *Να περιγράφεις την μορφή των διαδρόμων σε σύγχρονους υπολογιστές*

Σε ένα υπολογιστικό σύστημα τα επιμέρους υποσυστήματα πρέπει να διασυνδέονται μεταξύ τους για να ανταλλάσσουν δεδομένα. Για παράδειγμα, η κεντρική μνήμη επικοινωνεί με τον επεξεργαστή, όπως και οι περιφερειακές μονάδες. Αυτό γίνεται με την ύπαρξη του *διαδρόμου* (bus), δηλαδή ενός συνόλου παράλληλων καλωδίων στα οποία συνδέονται οι επιμέρους μονάδες. Το κύριο πλεονέκτημα του διαδρόμου είναι η



ευκολία με την οποία μπορούμε να τον επεκτείνουμε προσθέτοντας επιπλέον

περιφερειακές μονάδες. Απλά συνδέουμε την αντίστοιχη μονάδα σε μια κενή θέση, η οποία πρέπει να ακολουθεί το συγκεκριμένο πρωτόκολλο επικοινωνίας.

Με τον όρο *πρωτόκολλο επικοινωνίας διαδρόμου* (bus communication protocol) αναφερόμαστε στο σύνολο των κανόνων που καθορίζουν τον τρόπο με τον οποίο γίνεται η ανταλλαγή σημάτων και δεδομένων μεταξύ δυο περιφερειακών μονάδων που είναι διασυνδεδεμένες μέσω του διαδρόμου αυτού.

Ο διάδρομος αποτελείται από τις *γραμμές ελέγχου* (control) και τις *γραμμές δεδομένων* (data).

- ☑ Οι γραμμές ελέγχου χρησιμοποιούνται για τη μετάδοση των σημάτων ελέγχου, τα οποία καθορίζουν ποια μονάδα αποστέλλει δεδομένα, ποια μονάδα έχει τον έλεγχο των γραμμών δεδομένων, τι είδους δεδομένα μεταδίδονται κλπ.
- ☑ Οι γραμμές δεδομένων του διαδρόμου μεταφέρουν δυαδικές πληροφορίες από τη συσκευή-αποστολέα στη συσκευή-παραλήπτη. Οι δυαδικές πληροφορίες μπορεί να είναι εντολές προγράμματος, διευθύνσεις μνήμης ή ψηφιακά δεδομένα όπως για παράδειγμα περιεχόμενα του δίσκου.

Αν ο δίσκος θέλει να γράψει κάποια δεδομένα στη μνήμη, οι γραμμές δεδομένων θα περιέχουν στην αρχή τη διεύθυνση της μνήμης στην οποία θα γίνει η εγγραφή και κατόπιν τα δεδομένα του δίσκου. Στην αρχή οι γραμμές ελέγχου θα περιέχουν το σήμα ελέγχου που καθορίζει ότι στο διάδρομο δεδομένων περιέχεται διεύθυνση και στη συνέχεια ένα άλλο σήμα ελέγχου που θα δηλώνει ότι πλέον περιέχονται δεδομένα. Σε μερικές περιπτώσεις χρησιμοποιείται ένα «διευρυμένο» είδος διαδρόμου, όπου οι γραμμές διευθύνσεων είναι ξεχωριστές από τις γραμμές δεδομένων. Στις περισσότερες όμως περιπτώσεις δεδομένα και διευθύνσεις μεταφέρονται από το ίδιο σύνολο γραμμών με τα κατάλληλα σήματα ελέγχου να διαφοροποιούν την κάθε μετάδοση.

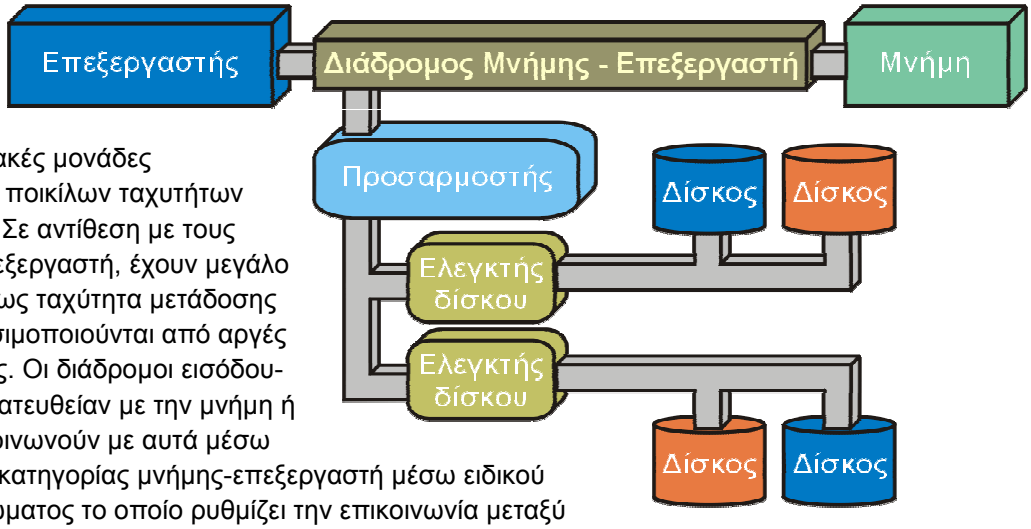
Τα κύρια χαρακτηριστικά ενός διαδρόμου είναι η *ταχύτητα* ή *συχνότητα μετάδοσης των bits* (bus clock, bandwith), το *εύρος* του (bus width) και ο *χρόνος αδράνειας* (bus latency).

- **Ταχύτητα διαδρόμου** ονομάζεται η ποσότητα των δυαδικών δεδομένων που μεταδίδονται σε μία μονάδα χρόνου. Η ταχύτητα μετριέται σε bits ανά δευτερόλεπτο. Ειδικά για διαδρόμους που η μετάδοση γίνεται σε προκαθορισμένους παλμούς ενός ρολογιού, αντί του όρου ταχύτητα συχνά χρησιμοποιούμε τον όρο *συχνότητα του διαδρόμου, η οποία* μετριέται σε Hz. Έτσι ένας διάδρομος με ρολόι 33 MHz σημαίνει ότι μπορεί να μεταδίδει σε κάθε γραμμή του 33 εκατομμύρια bits το δευτερόλεπτο.
- **Χρόνος αδράνειας** ονομάζεται ο χρόνος που χρειάζεται για τη μετάδοση του πρώτου τμήματος δεδομένων από το ένα σημείο του διαδρόμου στο άλλο, από το δίσκο στη κεντρική μνήμη, από την κεντρική μνήμη στη λανθάνουσα μνήμη κλπ. Ο χρόνος αδράνειας εξαρτάται από το μήκος του διαδρόμου μεταξύ των δύο σημείων. Όσο μεγαλύτερο είναι το μήκος, και όσο περισσότερες μονάδες παρεμβάλλονται στο διάδρομο, τόσο μεγαλύτερος είναι ο χρόνος αυτός.
- **Εύρος διαδρόμου** ονομάζεται το πλήθος των παράλληλων γραμμών που αποτελούν το διάδρομο. Το εύρος αντιπροσωπεύει τον αριθμό των bits που μεταδίδονται ταυτόχρονα και κυμαίνεται από 8 έως 64 ή και πολλές φορές 128

bits. Πάντα λόγω του μεγέθους των δεδομένων, το εύρος είναι πολλαπλάσιο του μήκους λέξης ενός υπολογιστή. Το εύρος του διαδρόμου επί την ταχύτητα του μας δίνουν το μέγιστο αριθμό των bytes που μεταδίδονται από όλα τα κανάλια του διαδρόμου, παράλληλα, και ονομάζεται *διαμεταγωγή* (throughput). Η τιμή αυτή είναι η μέγιστη δυνατή, καθώς στην πράξη ο όγκος των πληροφοριών που μεταδίδονται στη μονάδα του χρόνου είναι μικρότερος

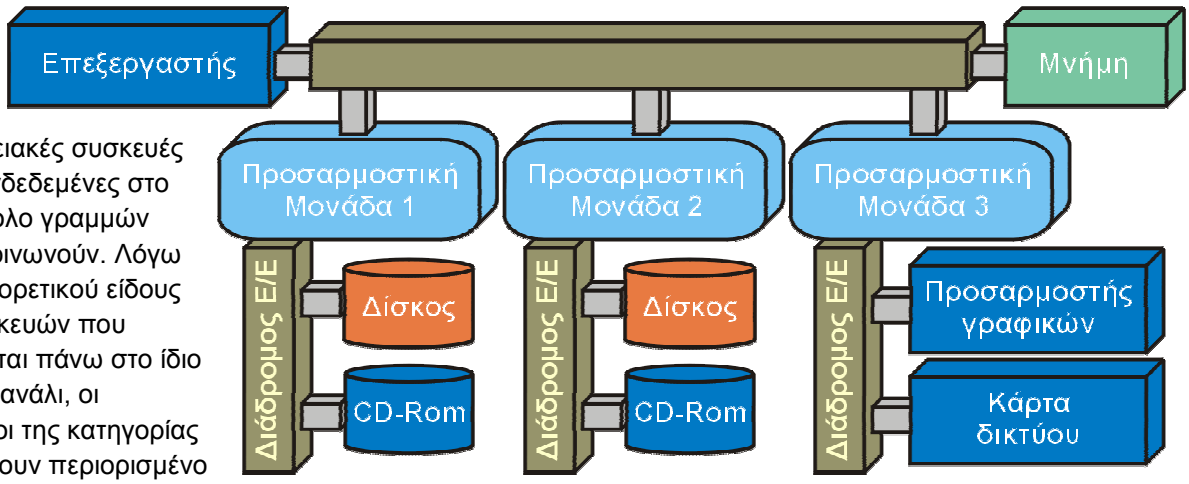
Οι διάδρομοι ταξινομούνται σε τρεις κατηγορίες: τους *διαδρόμους μνήμης-επεξεργαστή* (memory-CPU bus), τους *διαδρόμους εισόδου-εξόδου* (input-output bus) και τους *επίπεδους διαδρόμους* (flat bus).

Οι **διάδρομοι μνήμης-επεξεργαστή** έχουν μικρό μήκος, πολύ μεγάλες ταχύτητες μετάδοσης δεδομένων και ο χρονισμός τους εξαρτάται από την ταχύτητα ανάκτησης δεδομένων από τη μνήμη.



Οι **διάδρομοι εισόδου-εξόδου**, διασύνδεουν περιφερειακές μονάδες διαφορετικού τύπου και ποικίλων ταχυτήτων μετάδοσης δεδομένων. Σε αντίθεση με τους διαδρόμους μνήμης-επεξεργαστή, έχουν μεγάλο μήκος, και μικρή συνήθως ταχύτητα μετάδοσης δεδομένων καθώς χρησιμοποιούνται από αργές περιφερειακές συσκευές. Οι διάδρομοι εισόδου-εξόδου δε συνδέονται κατευθείαν με την μνήμη ή τον επεξεργαστή. Επικοινωνούν με αυτά μέσω ξεχωριστού διάδρομου κατηγορίας μνήμης-επεξεργαστή μέσω ειδικού προσαρμοστικού κυκλώματος το οποίο ρυθμίζει την επικοινωνία μεταξύ μονάδων με διαφορετική συχνότητα λειτουργίας.

Τέλος, οι **επίπεδοι διάδρομοι**, διασυνδέουν όλων των ειδών τις συσκευές στο ίδιο φυσικό κανάλι. Έτσι επεξεργαστής, μνήμη, υποσύστημα γραφικών, δίσκοι και άλλες



περιφερειακές συσκευές είναι συνδεδεμένες στο ίδιο σύνολο γραμμών και επικοινωνούν. Λόγω του διαφορετικού είδους των συσκευών που συνδέονται πάνω στο ίδιο φυσικό κανάλι, οι διάδρομοι της κατηγορίας αυτής έχουν περιορισμένο εύρος και ταχύτητες μετάδοσης. Συνήθως υπάρχουν ειδικά προσαρμοστικά κυκλώματα με σκοπό να προσαρμόζουν αργές περιφερειακές συσκευές καθώς και γρήγορες μονάδες όπως οι μνήμες ή ο επεξεργαστής.

Τα περισσότερα υπολογιστικά συστήματα, κυρίως οι υπερυπολογιστές, χρησιμοποιούν συνδυασμό και των τριών κατηγοριών διαδρόμων: Υπάρχει ένας διάδρομος μνήμης-επεξεργαστή που διασυνδέεται με έναν ή περισσότερους διάδρομους εισόδου-εξόδου μέσω ειδικών προσαρμοστών. Η διάταξη αυτή προσφέρει το πλεονέκτημα της γρήγορης επικοινωνίας επεξεργαστή και κεντρικής μνήμης, καθώς και επικοινωνία σταθερού ρυθμού για τις περιφερειακές μονάδες.

Διακοπές

Οι *διακοπές* (interrupts) είναι τα σήματα που παράγονται όταν χρειάζεται να πραγματοποιηθεί κάποια διαδικασία εισόδου/εξόδου και απαιτείται η χρησιμοποίηση της κεντρικής μονάδας επεξεργασίας. Οι διακοπές χωρίζονται σε *διακοπές υλικού* (hardware interrupts) και *διακοπές λογισμικού* (software interrupts), ανάλογα με την προέλευσή τους.

> Διακοπές υλικού συμβαίνουν κάθε φορά που πιέζεται ένα πλήκτρο ή μετακινείται το ποντίκι. Διακοπές λογισμικού παράγονται από ένα πρόγραμμα, όταν για παράδειγμα απαιτείται ανάγνωση ή εγγραφή στο δίσκο.

Ένας εσωτερικός χρονιστής διακόπτει την κεντρική μονάδα επεξεργασίας αρκετές φορές το δευτερόλεπτο, για να μοιράζει τον έλεγχο σε διαφορετικά προγράμματα σε συστήματα πολυεπεξεργασίας, για να κρατά την ώρα κ.ά. Όταν συμβαίνει μια διακοπή, ο έλεγχος μεταφέρεται στο λειτουργικό σύστημα, το οποίο και αναλαμβάνει να εξυπηρετήσει την αίτηση της διακοπής, εκτελώντας διάφορες λειτουργίες. Η αίτηση διακοπής περιγράφεται με το *διάνυσμα διακοπής* (interrupt vector). Αυτό είναι η διεύθυνση του προγράμματος που θα εκτελεστεί για να εξυπηρετηθεί η αίτηση της διακοπής. Αν κάποιο πρόγραμμα χρειάζεται να διαβάσει από το πληκτρολόγιο, αποστέλλεται το αντίστοιχο διάνυσμα διακοπής με διεύθυνση το «πρόγραμμα» του λειτουργικού συστήματος που διαβάζει από το πληκτρολόγιο. Ο επεξεργαστής στη συνέχεια εκτελεί αυτό το «πρόγραμμα», που ονομάζεται *ρουτίνα εξυπηρέτησης διακοπής* (interrupt service routine) και στο τέλος αποθηκεύει σε προκαθορισμένους καταχωρητές τα αποτελέσματα, ώστε το πρόγραμμα να τα χρησιμοποιήσει.

Οι διακοπές έχουν προτεραιότητες, ώστε συσκευές ή προγράμματα που έχουν υψηλότερη προτεραιότητα να εξυπηρετούνται κατά προτίμηση, διακόπτοντας (σε ορισμένα συστήματα ακόμα και στην μέση) την εξυπηρέτηση άλλων συσκευών. Για το σκοπό αυτό υπάρχουν διάφορα συστήματα που καθορίζουν τις προτεραιότητες των συσκευών στο διάδρομο περιφερειακών μονάδων.

> Σύμφωνα με το μοντέλο του διαδρόμου και των διακοπών που είδαμε, η μεταφορά δεδομένων από το δίσκο στην κεντρική μνήμη, γίνεται ως εξής:
>
> - Πραγματοποιείται μια αίτηση διακοπής για ανάγνωση από το δίσκο
> - Ο επεξεργαστής σταματά να εκτελεί ό,τι έκανε μέχρι εκείνη τη στιγμή
> - Για όσο διάστημα γίνεται ανάγνωση από το δίσκο και μετακίνηση στη μνήμη, ο επεξεργαστής εκτελεί τη ρουτίνα μετακίνησης δεδομένων.

Μονάδα Απευθείας Προσπέλασης Μνήμης (DMA)

Από το παράδειγμα στο τέλος της προηγούμενης ενότητας βλέπουμε ότι σημαντικό μέρος της ισχύος της μονάδας επεξεργασίας καταναλώνεται για την μετακίνηση μεγάλων όγκων δεδομένων από και προς το δίσκο. Η λύση είναι η ύπαρξη ενός ξεχωριστού επεξεργαστή, ο οποίος εξυπηρετεί κάθε αίτηση ανάγνωσης ή εγγραφής από και προς το δίσκο αντίστοιχα. Αυτός αναλαμβάνει τον έλεγχο του διαδρόμου και επομένως των μετακινήσεων, απελευθερώνοντας τον επεξεργαστή, ο οποίος μπορεί να κάνει άλλους υπολογισμούς, οι οποίοι δε χρειάζονται τον έλεγχο του διαδρόμου. Ο «επεξεργαστής» αυτός ονομάζεται *μονάδα απευθείας προσπέλασης μνήμης* (direct memory access unit) και αποτελεί ένα ξεχωριστό κύκλωμα, διασυνδεδεμένο στο διάδρομο, που μεταφέρει δεδομένα από τις περιφερειακές μονάδες στην κεντρική μνήμη χωρίς τη διαμεσολάβηση του κεντρικού επεξεργαστή.

Στους προσωπικούς υπολογιστές υπάρχουν οκτώ διαφορετικά κανάλια χρήσης της μονάδας απευθείας προσπέλασης μνήμης. Το κάθε κανάλι αντιπροσωπεύει διαφορετικά χαρακτηριστικά μεταφοράς δεδομένων και χρησιμοποιείται από την κατάλληλη συσκευή. Για παράδειγμα, το υποσύστημα ήχου χρησιμοποιεί το κανάλι 1 που μεταφέρει δεδομένα ανά 8 bits, ενώ ο δίσκος χρησιμοποιεί κανάλι εύρους 16 bit.

Ανακεφαλαίωση

Ο διάδρομος αποτελεί το μέσο διασύνδεσης μεταξύ των επιμέρους συστημάτων που αποτελούν τον υπολογιστή. Τα χαρακτηριστικά του είναι η ταχύτητα λειτουργίας του, το πλήθος των bits που μεταδίδονται ταυτόχρονα (εύρος) και ο χρόνος αδράνειας κατά τη μετάδοση. Οι κυριότερες κατηγορίες διαδρόμων είναι οι επίπεδοι διάδρομοι, οι διάδρομοι μνήμης-επεξεργαστή και οι διάδρομοι εισόδου-εξόδου.

Οι διακοπές αποτελούν τον τρόπο ελέγχου του διαδρόμου από κάποια συσκευή. Γίνεται αίτηση διακοπής, εξυπηρετείται η αίτηση από τον επεξεργαστή, και στη συνέχεια η συσκευή που ζήτησε τη διακοπή, επικοινωνεί με τον επεξεργαστή μέσω ενός συγκεκριμένου προγράμματος, της ρουτίνας εξυπηρέτησης διακοπής.

Η μονάδα απευθείας προσπέλασης μνήμης αναλαμβάνει την απευθείας επικοινωνία της κεντρικής μνήμης με κάποια περιφερειακή συσκευή, ώστε να απελευθερωθεί ο επεξεργαστής, ο οποίος μπορεί να εκτελεί άλλα προγράμματα.

Γλωσσάριο όρων

Γραμμές Δεδομένων	*Data*
Γραμμές Ελέγχου	*Control*
Διάδρομος	*Bus*
Διάδρομος Εισόδου-Εξόδου	*Input-Output Bus*
Διάδρομος Μνήμης-Επεξεργαστή	*Memory-Cpu Bus*
Διακοπή	*Interrupt*
Διακοπή Λογισμικού	*Software Interrupt*
Διακοπή Υλικού	*Hardware Interrupt*
Διάνυσμα Διακοπής	*Interrupt Vector*
Επίπεδος Διάδρομος	*Flat Bus*
Εύρος Διαδρόμου	*Bus Bandwidth*
Διαμεταγωγή	*Throughput*
Μονάδα Απευθείας Προσπέλασης Μνήμης	*DMA*
Πρωτόκολλο Επικοινωνίας Διαδρόμου	*Bus Communication Protocol*
Ρουτίνα Εξυπηρέτησης Διακοπής	*Interrupt Service Routine*
Ταχύτητα Διαδρόμου	*Bus Clock*
Χρόνος Αδράνειας	*Latency*

Ερωτήσεις

? Τι είναι διάδρομος;

? Ποια είναι τα χαρακτηριστικά ενός διαδρόμου;

? Ποιές είναι οι κατηγορίες διαδρόμων;

? Τι είναι DMA;

Μάθημα 5.2 Τεχνολογίες Εκτυπωτών

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι να περιγράψει τις σύγχρονες τεχνολογίες εκτύπωσης και τις διάφορες κατηγορίες εκτυπωτών που χρησιμοποιούνται από τους υπολογιστές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να απαριθμείς τις κατηγορίες των εκτυπωτών*
- *Να περιγράφεις τη λειτουργία του κρουστικού εκτυπωτή*
- *Να περιγράφεις τη λειτουργία του εκτυπωτή ψεκασμού μελάνης*
- *Να περιγράφεις τη λειτουργία του εκτυπωτή laser*
- *Να εξηγείς πώς λειτουργούν οι σχεδιογράφοι*

Ο *εκτυπωτής* (printer) είναι το μέσο στο οποίο αποτυπώνονται σε χαρτί τα δεδομένα που είναι αποθηκευμένα στον υπολογιστή. Ο εκτυπωτής τυπώνει κείμενα μορφοποιημένα από επεξεργαστές κειμένου, εικόνες και σχέδια από ειδικά προγράμματα επεξεργασίας εικόνων, αριθμητικά δεδομένα από λογιστικά φύλλα, αποδείξεις και τιμολόγια σε εφαρμογές επιχειρήσεων και ό,τι άλλο μπορεί να αναπαρασταθεί σε χαρτί.

Η εξέλιξη στην τεχνολογία εκτύπωσης είναι πολύ μεγάλη, ιδιαίτερα την τελευταία δεκαετία. Πριν από μερικά χρόνια για τους χρήστες προσωπικών υπολογιστών ήταν σχεδόν αδύνατη η εκτύπωση γραφικών και σύνθετων μορφών κειμένου με εικόνες. Όμως, οι εξελίξεις στις μηχανές εκτύπωσης που βασίζονται στην τεχνολογία των laser, έχουν κάνει δυνατή την αναπαραγωγή και των πιο δύσκολων εκτυπώσεων. Σήμερα οι εκτυπωτές κοινής χρήσης προσεγγίζουν σε ποιότητα επαγγελματικές φωτοστοιχειοθετικές μηχανές.

Οι εκτυπωτές διαχωρίζονται σε τρεις κατηγορίες ανάλογα με την τεχνολογία εκτύπωσης που χρησιμοποιούν:

- **Κρουστικοί εκτυπωτές** (Impact printers)
- **Εκτυπωτές ψεκασμού** (Inkjet printers)
- **Εκτυπωτές laser** (Laser printers).

Ένας άλλος διαχωρισμός των εκτυπωτών γίνεται ως προς τον τρόπο εκτύπωσης, σε εκτυπωτές γραμμής και εκτυπωτές σελίδας.

- Οι *εκτυπωτές σελίδας* (page printers) ολοκληρώνουν το σχηματισμό της σελίδας στην μνήμη τους πριν προχωρήσουν στην εκτύπωση και κατόπιν την εκτυπώνουν.
- Οι *εκτυπωτές γραμμής* (line printers) εκτυπώνουν τη σελίδα γραμμή-γραμμή.

Τα βασικότερα χαρακτηριστικά των εκτυπωτών είναι τα εξής:

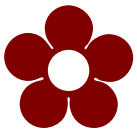
Ταχύτητα εκτύπωσης: Ανάλογα με την τεχνολογία εκτύπωσης έχουμε και διαφορετικές μονάδες μέτρησης. Στους εκτυπωτές γραμμής, η ταχύτητα εκφράζεται σε χαρακτήρες ανά δευτερόλεπτο (characters per second - cps) ενώ στους εκτυπωτές σελίδας, σε σελίδες ανά λεπτό (pages per minute - ppm).

Ανάλυση εκτύπωσης: Η ελάχιστη μονάδα εκτύπωσης (με άλλα λόγια, το πιο μικρό ίχνος) που μπορεί να τυπωθεί στο χαρτί και ονομάζεται *κουκίδα* (dot). Η ανάλυση εκτύπωσης ορίζεται ώς το πλήθος των διαφορετικών κουκίδων που μπορούν να εκτυπωθούν ανά ίντσα[1] (dots per inch - dpi).

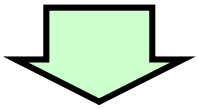
Η ποιότητα της τελικής εκτύπωσης ενός εκτυπωτή εξαρτάται σε μεγάλο βαθμό από την ανάλυση εκτύπωσης που αναφέρει ο κατασκευαστής του, όχι όμως αποκλειστικά από αυτήν. Στην πράξη η τεχνολογία που χρησιμοποιείται για την εκτύπωση καθορίζει και την ποιότητα, όπως θα δούμε στη συνέχεια. Το πλήθος των κουκίδων ανά ίντσα είναι σημαντικό, όμως παίζουν ρόλο και άλλα χαρακτηριστικά, όπως το μέγεθος και το σχήμα της κουκίδας, η ακρίβεια στοίχισης των κουκίδων που ανήκουν στην ίδια ευθεία, το χαρτί εκτύπωσης και άλλα. Έτσι η επιλογή ενός ποιοτικού εκτυπωτή, εξαρτάται από πολλούς άλλους παράγοντες εκτός της τιμής της ανάλυσης εκτύπωσης.

Κρουστικοί Εκτυπωτές

Οι κρουστικοί εκτυπωτές αποτελούν την πιο παλιά κατηγορία εκτυπωτών. Η τεχνολογία εκτύπωσης τους βασίζεται στην ύπαρξη μιας κινούμενης κεφαλής, η οποία περιέχει ακίδες ή χαρακτήρες που κτυπάνε μία μελανοταινία προς το χαρτί. Έτσι σε κάθε κτύπημα αποτυπώνεται στο χαρτί ένας συγκεκριμένος χαρακτήρας ή ένα ίχνος μελανιού. Λειτουργούν με πανομοιότυπο τρόπο όπως οι παλαιές γραφομηχανές, όμως εδώ τα ηλεκτρονικά κυκλώματα και οι μηχανισμοί του εκτυπωτή καθορίζουν την μετακίνηση του χαρτιού. Η ταχύτητα εκτύπωσης στους κρουστικούς εκτυπωτές μετριέται και σε γραμμές ανά δευτερόλεπτο (lines per second – lps).



Εκτυπωτές μαργαρίτας



Εκτυπωτές πίνακα ακίδων

Η πρώτη μορφή κρουστικού εκτυπωτή, ήταν ο *εκτυπωτής μαργαρίτας* (daisy wheel). Αυτός, όπως και οι γραφομηχανές, περιείχε έναν τροχό, στον οποίο υπήρχαν σε διάταξη μαργαρίτας οι χαρακτήρες μιας γραμματοσειράς. Αν ο χρήστης ήθελε να αλλάξει γραμματοσειρά, σταματούσε την εκτύπωση, άλλαζε μαργαρίτα χαρακτήρων, και συνέχιζε. Είναι φανερό ότι οι εκτυπωτές αυτοί δεν μπορούσαν να απεικονίσουν γραφικά ή να τυπώσουν σε οποιοδήποτε σημείο του χαρτιού.

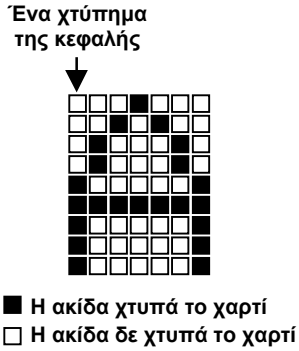
Σύντομα η τεχνολογία μαργαρίτας στους κρουστικούς εκτυπωτές ξεπεράστηκε και κυριάρχησε το μοντέλο των *εκτυπωτών πίνακα ακίδων* (dot matrix). Εδώ υπάρχει μια κινούμενη κεφαλή που περιέχει μικροσκοπικές μεταλλικές ακίδες, πολύ κοντά τη μία στην άλλη, διατεταγμένες σε μια ή δυο σειρές. Ανάμεσα στις ακίδες και το χαρτί υπάρχει η μελανοταινία. Το κτύπημα των ακίδων πάνω στη μελανοταινία, αφήνει το ίχνος τους στο χαρτί. Έτσι οι χαρακτήρες αποτυπώνονται με το συνδυασμό διαδοχικών κτυπημάτων σε προκαθορισμένες θέσεις, που αντιστοιχούν στον κάθε χαρακτήρα. Όσο πιο πολλά κτυπήματα χρησιμοποιούνται για την απεικόνιση ενός χαρακτήρα τόσο πιο ποιοτική εκτύπωση παράγεται στο χαρτί. Θεωρητικά η ποιότητα της εκτύπωσης δεν εξαρτάται τόσο από το πλήθος των ακίδων της κεφαλής, όσο από το πόσες ακίδες χρησιμοποιούνται ταυτόχρονα για την εκτύπωση του ίδιου χαρακτήρα.

---

[1] Ίντσα: αγγλοσαξονική μονάδα μήκους ίση με 2,54 εκ.

Μπορούμε να πετύχουμε το ίδιο ποιοτικό αποτέλεσμα με κεφαλές των 24 και 9 ακίδων, όμως στη δεύτερη περίπτωση χρειάζονται πολλά περάσματα για κάθε χαρακτήρα, προκαλώντας έτσι καθυστέρηση στην εκτύπωση.

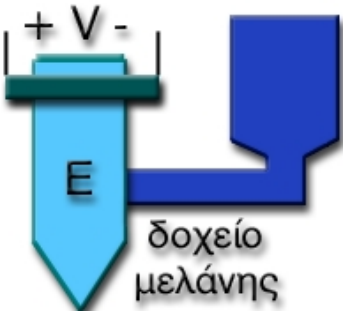
Στην περίπτωση των 9 ακίδων, επειδή η πυκνότητα των ακίδων είναι μικρότερη, η κεφαλή μετατοπίζεται ελάχιστα σε κάθε πέρασμα, ώστε να καλύπτει τα κενά της εκτύπωσης. Στην πράξη η διαδικασία αυτή είναι αρκετά σύνθετη: απαιτεί το συγχρονισμό της κεφαλής με το μηχανισμό τροφοδοσίας χαρτιού ώστε να γίνει με ακρίβεια η μικρή μετατόπιση του χαρτιού που χρειάζεται για το επόμενο πέρασμα.



Στον ίδιο λόγο οφείλονται και τα προβλήματα που εμφανίζουν οι εκτυπωτές ακίδων στην εκτύπωση γραμμικών σχεδίων. Οι οριζόντιες γραμμές τυπώνονται αρκετά εύκολα και με ακρίβεια, αφού γίνεται μόνο οριζόντια μετακίνηση των ακίδων εκτύπωσης κατά μήκος του χαρτιού. Στις κατακόρυφες και πολύ περισσότερο στις διαγώνιες γραμμές όμως τα πράγματα δεν είναι το ίδιο εύκολα. Απαιτούνται αργές μετακινήσεις της κεφαλής για να αποφεύγονται τα κενά και οι ασυνέχειες. Σε εκτυπώσεις τέτοιου είδους υπερτερούν οι εκτυπωτές με πολλές ακίδες διατεταγμένες σε δύο ή τρεις στήλες. Αυτοί με λιγότερα περάσματα μπορούν να επιτύχουν μερική επικάλυψη των κουκίδων και επομένως την εικόνα συνεχούς κατακόρυφης γραμμής.

Εκτυπωτές ψεκασμού μελάνης

Οι εκτυπωτές ψεκασμού μελάνης αποτελούν το ενδιάμεσο στάδιο μεταξύ των κρουστικών εκτυπωτών και των εκτυπωτών laser. Συνδυάζουν τη χαμηλή τιμή των κρουστικών εκτυπωτών με την ταχύτητα και την ποιότητα εκτύπωσης των ακριβότερων εκτυπωτών laser.



Όπως και οι κρουστικοί εκτυπωτές, οι εκτυπωτές ψεκασμού μελάνης περιέχουν μία κεφαλή εκτύπωσης που κινείται οριζόντια πάνω στο χαρτί σχηματίζοντας την εκτύπωση γραμμή-γραμμή. Η κεφαλή αυτή όμως δεν αποτελείται από ακίδες αλλά από μικροσκοπικές τρύπες, τα ακροφύσια, που εκτοξεύουν απειροελάχιστες ποσότητες μελάνης στο χαρτί, δημιουργώντας κουκίδες.

Η κεφαλή περιέχει ένα σύνολο θαλάμων μελάνης που καταλήγουν στα ακροφύσια. Το μελάνι θερμαίνεται σε κάθε θάλαμο, μέσω μιας αντίστασης. Η αυξημένη θερμότητα, δημιουργεί σε ελάχιστο χρόνο μία φυσαλίδα αερίου, η οποία σπρώχνει μία μικρή ποσότητα μελάνης προς τα έξω, μέσω του ακροφυσίου. Μερικές κεφαλές δεν έχουν θερμαντική αντίσταση, αλλά κρύσταλλο χαλαζία, ο οποίος ταλαντώνεται με την παροχή ηλεκτρικού ρεύματος ασκώντας πίεση στο θάλαμο με αποτέλεσμα την εκτόξευση της μελάνης.

Το μεγαλύτερο μειονέκτημα των εκτυπωτών ψεκασμού είναι η ακρίβεια του σχήματος της κουκίδας εκτύπωσης. Επειδή γίνεται έκχυση υγρής μελάνης απευθείας πάνω στο χαρτί, όταν χρησιμοποιούνται απορροφητικά χαρτιά γίνεται διάχυση της κουκίδας, με αποτέλεσμα η ευκρίνεια της εκτύπωσης να μειώνεται. Συνήθως για ποιοτικές εκτυπώσεις χρησιμοποιούνται ειδικού τύπου χαρτιά, στα οποία η μελάνη στεγνώνει στην επιφάνεια χωρίς να διαχέεται. Από την άλλη πλευρά, λόγω της ύπαρξης θαλάμων μελάνης σε υγρή μορφή, είναι δυνατή η δημιουργία πολύ ποιοτικών έγχρωμων εκτυπώσεων με εύκολη ανάμειξη των μελανιών βασικών χρωμάτων. Στις μέρες μας που το κόστος των μονόχρωμων εκτυπωτών laser έχει πέσει κατακόρυφα, πλησιάζοντας το κόστος των εκτυπωτών ψεκασμού, η τεχνολογία έκχυσης μελάνης

χρησιμοποιείται κυρίως για έγχρωμες εκτυπώσεις, με ποιότητα κατάλληλη ακόμα και για φωτογραφίες.

Εκτυπωτές laser

Η χρήση της τεχνολογίας laser έχει επιφέρει κατακόρυφη βελτίωση στους εκτυπωτές των υπολογιστών. Οι εκτυπωτές laser ανήκουν στην κατηγορία των εκτυπωτών σελίδας. Δεν κάνουν σάρωση της σελίδας γραμμή προς γραμμή· ο υπολογιστής στέλνει τα δεδομένα για κάθε σελίδα, που αποθηκεύονται στη μνήμη του εκτυπωτή. Εκεί σχηματίζεται η τελική εικόνα της σελίδας, κουκίδα προς κουκίδα, και κατόπιν αρχίζει η διαδικασία της εκτύπωσής της.

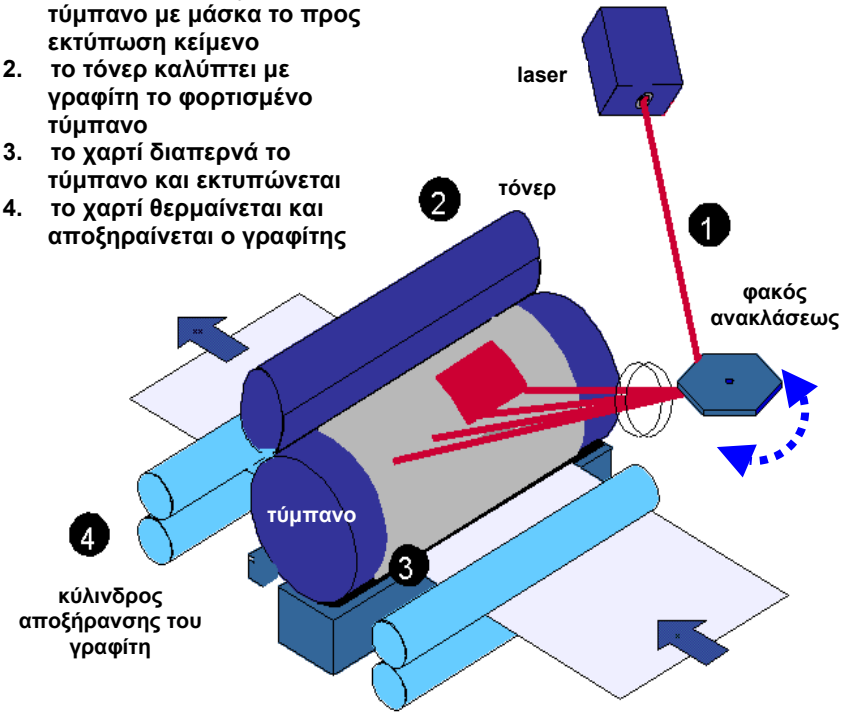
Οι σύγχρονοι εκτυπωτές laser έχουν ονομαστικές ταχύτητες που πλησιάζουν τις 16 και 20 σελίδες το λεπτό σε αναλύσεις των 600 ή και 1200 dpi. Η ονομαστική ταχύτητα εκτύπωσης αφορά την παραγωγή πολλαπλών αντιγράφων της ίδιας σελίδας. Στις συνήθεις εκτυπωτικές εργασίες, χρειάζεται ένας αρχικός χρόνος προετοιμασίας της κάθε σελίδας στη μνήμη, επομένως η πραγματική ταχύτητα εκτύπωσης είναι μικρότερη της ονομαστικής ταχύτητας που αναφέρει κάθε κατασκευαστής.

Ο μηχανισμός εκτύπωσης είναι παρόμοιος με αυτόν των φωτοαντιγραφικών μηχανημάτων. Περιέχει ένα φωτοευαίσθητο κύλινδρο (*τύμπανο*, drum), πάνω στον οποίο σχηματίζεται η εικόνα της σελίδας, με τη βοήθεια μιας ακτίνας laser. Αναλυτικά, η διαδικασία εκτύπωσης περιλαμβάνει τα εξής στάδια:

Αρχικά, σχηματίζεται η ακριβής εικόνα της σελίδας στη μνήμη του εκτυπωτή.

☑ Με βάση τη μορφή της σελίδας στη μνήμη, μια ακτίνα laser (μονοχρωματική δέσμη φωτός, μεγάλης ισχύος) πέφτει πάνω στο τύμπανο. Αυτό επειδή είναι φωτοευαίσθητο έχει την δυνατότητα αποφορτίσεως σε όποια σημεία πέσει φωτεινή ακτινοβολία και έτσι φορτίζεται με θετικό στατικό ηλεκτρισμό δημιουργώντας το αρνητικό της σελίδας επάνω στην επιφάνειά του.

1. **το laser εκφορτίζει το τύμπανο με μάσκα το προς εκτύπωση κείμενο**
2. **το τόνερ καλύπτει με γραφίτη το φορτισμένο τύμπανο**
3. **το χαρτί διαπερνά το τύμπανο και εκτυπώνεται**
4. **το χαρτί θερμαίνεται και αποξηραίνεται ο γραφίτης**



☑ Στη συνέχεια το τύμπανο περιστρέφεται γύρω από ένα ειδικό δοχείο με γραφίτη που ονομάζεται toner. Ο γραφίτης, που είναι σε μορφή σκόνης, προσκολλάται στις φορτισμένες επιφάνειες του τυμπάνου, με αποτέλεσμα να δημιουργηθεί στη επιφάνεια του η εικόνα της σελίδας με γραφίτη.

☑ Ο μηχανισμός τροφοδοσίας χαρτιού προωθεί ένα λευκό χαρτί, το οποίο έρχεται σε άμεση επαφή με το τύμπανο, με αποτέλεσμα ο γραφίτης να μείνει πάνω στο χαρτί.

☑ Τέλος, το χαρτί με τη σχηματισμένη σελίδα, περνά από το πιεστήριο, μια συστοιχία κυλίνδρων που πιέζουν και ξεραίνουν το γραφίτη σε υψηλή θερμοκρασία ώστε να σταθεροποιηθεί στο χαρτί.

☑ Η επιφάνεια του τύμπανου εκφορτίζεται, καθαρίζεται από το γραφίτη, και συνεχίζεται η εκτύπωση των επόμενων αντιγράφων.

Ο υπολογιστής στέλνει στους εκτυπωτές laser την περιγραφή κάθε σελίδας, μαζί με οδηγίες για τον τρόπο εκτύπωσής της. Τα δεδομένα αυτά περιγράφονται με τη βοήθεια μιας ειδικής γλώσσας για την περιγραφή σελίδων. Οι κυριώτερες γλώσσες είναι η PCL (Printer Control Language) και η γλώσσα Postscript.

Σχεδιογράφοι (Plotters)

Οι *σχεδιογράφοι* (plotters), είναι ειδικές εκτυπωτικές μηχανές για γραμμικά σχέδια μεγάλων διαστάσεων, όπως είναι τα αρχιτεκτονικά, τα μηχανολογικά, τα ηλεκτρολογικά σχέδια κ.ά. Αρκετοί σχεδιογράφοι παράγουν αντίγραφα μεγέθους μέχρι και 40×48 ίντσες (περίπου 102×123 εκ.), πολύ μεγαλύτερα από τα συνήθη τυποποιημένα μεγέθη A3 και A4 που παράγουν οι εκτυπωτές γραφείου. Οι σχεδιογράφοι ταξινομούνται με βάση τον τρόπο δημιουργίας του σχεδίου. Έτσι υπάρχουν *σχεδιογράφοι μελάνης* και ηλεκτροστατικοί.

Οι **σχεδιογράφοι μελάνης** παράγουν σχέδια με την μετακίνηση ειδικών στυλό μελάνης πάνω στην επιφάνεια του χαρτιού. Διακρίνονται σε επίπεδους και κυλινδρικούς. Στους επίπεδους σχεδιογράφους μελάνης το χαρτί απλώνεται πάνω σε μια λεία επιφάνεια. Κατά μήκος του χαρτιού κινείται ένα σύστημα από στυλό μελάνης (1-4 στυλό διαφορετικών χρωμάτων). Στους κυλινδρικούς σχεδιογράφους, τα στυλό είναι σταθερά και κινείται το χαρτί πάνω σε ένα κύλινδρο, τυλιγμένο σε ρολό.

Οι **ηλεκτροστατικοί σχεδιογράφοι** βασίζονται στην ηλεκτροστατική φόρτιση του χαρτιού στα σημεία που θέλουμε, και μετά στην «εμφάνισή» του, δηλαδή στο πέρασμά του από γραφίτη ώστε να δημιουργηθεί το σχέδιο. Η τεχνική αυτή μοιάζει αρκετά με εκείνη των εκτυπωτών laser.

Ανακεφαλαίωση

Η εξέλιξη στις τεχνολογίες εκτύπωσης είναι ραγδαία, επιτρέποντας σε πολλές περιπτώσεις εκτυπώσεις επαγγελματικής ποιότητας. Διακρίνονται δύο είδη εκτυπωτών ανάλογα με τον τρόπο επεξεργασίας της εκτύπωσης: Εκτυπωτές γραμμής και εκτυπωτές σελίδας.

Τα βασικότερα χαρακτηριστικά των εκτυπωτών είναι η ταχύτητα και η ανάλυση εκτύπωσης. Ως προς την τεχνολογία που χρησιμοποιείται οι εκτυπωτές διακρίνονται σε κρουστικούς εκτυπωτές, εκτυπωτές ψεκασμού μελάνης και εκτυπωτές laser. Οι κρουστικοί εκτυπωτές τυπώνουν πιέζοντας την μελανοταινία προς το χαρτί, χρησιμοποιώντας πιο συχνά ακίδες (εκτυπωτές ακίδας) ή παλαιότερα έτοιμους χαρακτήρες (εκτυπωτές μαργαρίτας) όπως οι γραφομηχανές. Οι εκτυπωτές μελάνης βασίζονται στην έκχυση μελάνης σε μικροσκοπικές κουκίδες. Αποτελούν την καλύτερη λύση για φθηνές και ποιοτικές έγχρωμες εκτυπώσεις. Οι εκτυπωτές laser παράγουν τις πιο ποιοτικές εκτυπώσεις και λειτουργούν με ηλεκτροστατική φόρτιση του τυμπάνου, το οποίο αφού απορροφήσει σε επιλεγμένα σημεία γραφίτη, τον αποτυπώνει στο χαρτί.

Οι σχεδιογράφοι αποτελούν ειδική κατηγορία εκτυπωτών για την απεικόνιση αρχιτεκτονικών, μηχανολογικών και άλλων σχεδίων μεγάλων διαστάσεων.

Γλωσσάρι όρων

Γλώσσα Postscript	
Γλώσσα Printer Control Language	*PCL*
Γλώσσα Περιγραφής Σελίδας	*Page Description Language*
Γραμμές Ανά Δευτερόλεπτο	*Lines Per Second – lps*
Γραφίτης	*Toner*
Εκτυπωτής	*Printer*
Εκτυπωτής Laser	*Laser Printer*
Εκτυπωτής Ακίδας	*Dot Matrix Printer*
Εκτυπωτής Γραμμής	*Line Printer*
Εκτυπωτής Μαργαρίτας	*Daisy Wheel Printer*
Εκτυπωτής Σελίδας	*Page Printer*
Εκτυπωτής Ψεκασμού Μελάνης	*Inkjet Printer*
Ηλεκτροστατικός Σχεδιογράφος	
Κουκίδα	*Dot*
Σελίδες Ανά Λεπτό	*Pages Per Minute – Ppm*
Σχεδιογράφος	*Plotter*
Σχεδιογράφος Μελάνης	
Ταχύτητα Εκτύπωσης	
Τύμπανο	*Drum*
Χαρακτήρες Ανά Δευτερόλεπτο	*Characters Per Second - Cps*

Ερωτήσεις

? Ποια είναι τα κύρια χαρακτηριστικά των εκτυπωτών;

? Ποια είναι τα κύρια είδη των εκτυπωτών ανάλογα με την τεχνολογία εκτύπωσης που χρησιμοποιούν;

? Ποια είναι τα μειονεκτήματα των εκτυπωτών ψεκασμού;

? Τι είναι ο σχεδιογράφος;

Μάθημα 5.3 Τεχνολογία Απεικονιστικών Μονάδων

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει τις πιο αντιπροσωπευτικές τεχνολογίες που χρησιμοποιούνται στις οθόνες των υπολογιστών, τα χαρακτηριστικά και τον τρόπο λειτουργίας των οθονών καθοδικού σωλήνα και υγρών κρυστάλλων.

Τι θα μάθεις;

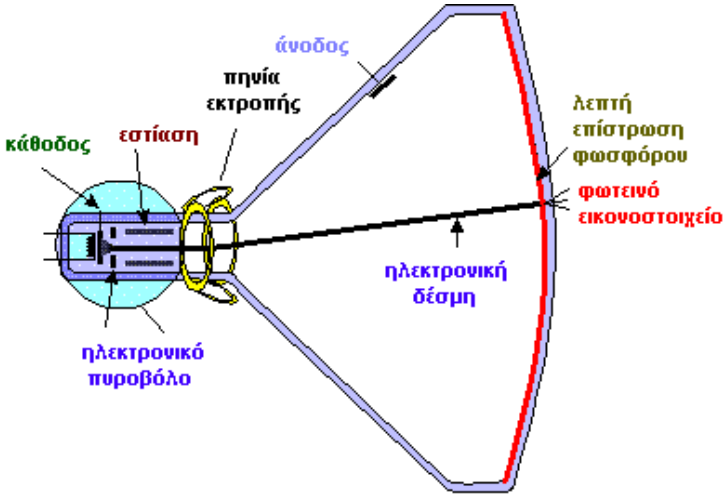
Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις τη λειτουργία της οθόνης καθοδικού σωλήνα*
- *Να εξηγείς την ανάλυση οθόνης, τη συχνότητα ανανέωσης και το βήμα κουκίδας*
- *Να αναλύεις τη λειτουργία των οθονών υγρών κρυστάλλων*

Η *οθόνη* (screen) αποτελεί το βασικότερο μέσο επικοινωνίας του υπολογιστή με το χρήστη. Εκεί απεικονίζονται τα αποτελέσματα κάθε λειτουργίας του υπολογιστή. Η τεχνολογία των οθονών που χρησιμοποιούνται στους ηλεκτρονικούς υπολογιστές είναι πολύ καλύτερη των κλασικών τηλεοράσεων, καθώς οι απαιτήσεις σε ευκρίνεια, καθαρότητα και ακρίβεια γεωμετρίας είναι μεγαλύτερες. Οι κατηγορίες οθονών που έχουν επικρατήσει στο χώρο των υπολογιστών είναι οι οθόνες καθοδικού σωλήνα και οι οθόνες υγρών κρυστάλλων.

Οθόνη καθοδικού σωλήνα

Οι *οθόνες καθοδικού σωλήνα* (Cathode Ray Tube - CRT) αποτελούνται από ένα (για μονόχρωμες οθόνες) ή τρία (για έγχρωμες οθόνες) πυροβόλα ηλεκτρονίων μέσα σε ένα σωλήνα. Τα ηλεκτρόνια, αφού διανύσουν το σωλήνα, προσκρούουν στο πίσω μέρος της οθόνης που είναι επιστρωμένο με φωσφορίζουσα ουσία. Το σημείο από όπου ξεκινά η ακτίνα των ηλεκτρονίων λέγεται *κάθοδος* (cathode). Στο σημείο πρόσκρουσης με την οθόνη, η φωσφορίζουσα ουσία διεγείρεται και λάμπει με αποτέλεσμα την εμφάνιση μιας φωτεινής κουκίδας. Η κουκίδα αυτή αποτελεί το ελάχιστο ίχνος απεικόνισης και ονομάζεται *εικονοστοιχείο* (pixel - picture element). Η διέγερση της φωσφορίζουσας ουσίας διαρκεί μερικά χιλιοστά του δευτερολέπτου, που αποτελούν το *χρόνο αναλαμπής* (persistence).



Μονόχρωμος Καθοδικός Σωλήνας

Για το σχηματισμό της εικόνας στην οθόνη,

απαιτείται η αναλαμπή πολλών τέτοιων στοιχειωδών κουκίδων φωσφόρου, άρα σχεδόν ταυτόχρονη διέγερση από το ηλεκτρονικό πυροβόλο. Για το λόγο αυτό το ηλεκτρονικό πυροβόλο δε στοχεύει προς μια συγκεκριμένη κατεύθυνση, αλλά με την βοήθεια πηνίων εκτροπής, βάλλει προς όλες τα σημεία της οθόνης για να σχηματιστεί η ζητούμενη εικόνα. Η σάρωση της οθόνης από το πυροβόλο γίνεται οριζόντια, γραμμή προς γραμμή, από το πάνω μέχρι το κάτω μέρος της, οπότε και ξαναρχίζει από την αρχή. Η διαδρομή αυτή πρέπει να πραγματοποιείται σε χρόνο ταχύτερο από του χρόνου αναλαμπής της φωσφορίζουσας ουσίας, ώστε το μάτι να βλέπει ταυτόχρονα ολόκληρη τη σχηματισμένη εικόνα.

Συχνότητες σάρωσης

- ⇨ Ο ρυθμός σάρωσης της οθόνης ανά γραμμές ονομάζεται *οριζόντια συχνότητα σάρωσης* (horizontal scanning frequency) και υποδηλώνει το πλήθος των γραμμών που εμφανίζονται ανά δευτερόλεπτο.
- ⇨ Ο αριθμός των διαφορετικών «οθονών» που μπορούν να εμφανιστούν ανά δευτερόλεπτο, δηλαδή το πλήθος των φορών που σχεδιάζεται ολόκληρη η οθόνη σε ένα δευτερόλεπτο, ονομάζεται *κατακόρυφη συχνότητα* (vertical scanning frequency), ή *συχνότητα ανανέωσης πλαισίου* (refresh rate).

Θεωρητικά το γινόμενο της κατακόρυφης συχνότητας επί το πλήθος των γραμμών στις οποίες είναι διαιρεμένη η οθόνη μας δίνει την οριζόντια συχνότητα. Λόγω όμως της αδράνειας των ηλεκτρονικών πυροβόλων, κυρίως κατά την μετακίνηση από την κάτω δεξιά γωνία στην πάνω αριστερή, η οριζόντια συχνότητα πρέπει να είναι λίγο μεγαλύτερη από το γινόμενο αυτό.

Μονόχρωμες οθόνες

Στις μονόχρωμες οθόνες υπάρχει ένα μοναδικό ηλεκτρονικό πυροβόλο. Όταν η ηλεκτρονική δέσμη προσπίπτει πάνω στην επιφάνεια της φωσφορίζουσας ουσίας δημιουργείται ένα λευκό ίχνος, αλλιώς η επιφάνεια παραμένει μαύρη. Στις πρώτες μονόχρωμες οθόνες κειμένου είχαμε μόνο τα δύο χρώματα, το λευκό και το μαύρο. Αργότερα οι οθόνες άρχισαν να απεικονίζουν γραφικά με διαφορετικές αποχρώσεις του γκρίζου, συνήθως 64 ή 256. Η ισχύς της ηλεκτρονικής ακτίνας καθόριζε και την ένταση της ακτινοβολούμενης λάμψης της φωσφορίζουσας ουσίας που αντιστοιχούσε σε διάφορα επίπεδα του του γκρίζου.

Έγχρωμες οθόνες

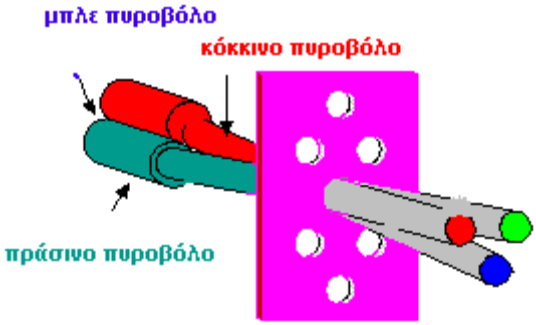
Η ίδια περίπου τεχνική χρησιμοποιείται και για τις έγχρωμες οθόνες. Αυτές περιέχουν τρία πυροβόλα, ένα για κάθε βασικό χρώμα (Κόκκινο/Πράσινο/Μπλέ – Red/Green/Blue - RGB).

Στη επιφάνεια δημιουργούνται ξεχωριστά ίχνη για κάθε χρώμα, πολύ κοντά το ένα στο άλλο, ώστε με την διέγερσή τους από τα αντίστοιχα πυροβόλα να συνδυάζονται και να δημιουργούν το εικονοστοιχείο, το οποίο φαίνεται έγχρωμο στο ανθρώπινο μάτι.

Η ισχύς του καθενός από τα τρία πυροβόλα καθορίζει τη *φωτεινότητα* (brightness) του αντίστοιχου βασικού χρώματος στο συνδυασμό για την παραγωγή του χρώματος του εικονοστοιχείου. Με τον τρόπο αυτό μπορούν να εμφανιστούν πολλά διαφορετικά χρώματα.

> Αν σε μια οθόνη για κάθε χρώμα υπάρχουν 256 στάθμες φωτεινότητας, τότε η οθόνη μπορεί να εμφανίσει $256 \times 256 \times 256 = 2^{24}$ διαφορετικά χρώματα (περίπου 17 εκατομμύρια).

Για να εξασφαλιστεί ότι τα τρία ηλεκτρονικά πυροβόλα θα προσπίπτουν πάνω στα σωστά εικονοστοιχεία, μεταξύ των πυροβόλων και της επιφάνειας της οθόνης παρεμβάλλεται μια μεταλλική πλάκα στην οποία έχουν ανοιχτεί τρύπες με σταθερή απόσταση μεταξύ τους. Η πλάκα αυτή ονομάζεται *μάσκα* (mask) και η απόσταση μεταξύ των τρυπών ονομάζεται *βήμα κουκίδας* (dot pitch) Το βήμα κουκίδας καθορίζει και τη μέγιστη ανάλυση, δηλαδή το πλήθος των διαφορετικών εικονοστοιχείων που μπορεί να απεικονίσει μια οθόνη συγκεκριμένου μεγέθους.



Το μέγεθος της οθόνης χαρακτηρίζεται από το μήκος της διαγωνίου της και μετράται σε ίντσες. Οι κατασκευαστές δίνουν συνήθως το μέγεθος της ονομαστικής διαγωνίου· αυτό είναι λίγο μεγαλύτερο από το πραγματικό μέγεθος του πλαισίου της οθόνης που είναι ορατό για τον άνθρωπο.

> Μια οθόνη ονομαστικής διαγωνίου 15 ιντσών (περίπου 38 εκ.) έχει ορατό πλαίσιο με διαστάσεις 30×23 cm και βήμα κουκίδας 0,25 mm.
>
> Κατά την οριζόντια διάσταση μπορεί να απεικονίσει 30cm / 0,25mm = 1200 διαφορετικά εικονοστοιχεία
>
> Κατά την κατακόρυφη διάσταση μπορεί να απεικονίσει 23cm / 0,25mm = 920 διαφορετικά εικονοστοιχεία.

Τα μεγέθη αυτά ορίζουν την μέγιστη *ανάλυση* (resolution) που μπορεί να απεικονίσει η συγκεκριμένη οθόνη. Η εικόνα που εμφανίζει ο υπολογιστής στην οθόνη, έχει διαστάσεις μικρότερες από τη μέγιστη ανάλυση που μπορεί να υποστηρίξει η οθόνη. Οι διαστάσεις απεικόνισης είναι τυποποιημένες: υπάρχουν αναλύσεις 640×480, 800×600, 1024×768 κλπ.

Από τη μεριά του υπολογιστή υπάρχει το υποσύστημα γραφικών, το οποίο καθορίζει την ανάλυση και το πλήθος των χρωμάτων της εικόνας που θα απεικονιστεί στην οθόνη. Στο υποσύστημα γραφικών υπάρχει ειδική μνήμη, στην οποία φυλάσσεται για κάθε εικονοστοιχείο η τιμή της φωτεινότητας του χρώματος ή των χρωμάτων.

> Αν το πλαίσιο της οθόνης έχει διαστάσεις 640×480 και χρησιμοποιούνται 24 bits για κάθε εικονοστοιχείο (8 bits για καθένα από τα τρία βασικά χρώματα), τότε η μνήμη του υποσυστήματος γραφικών έχει μέγεθος
>
> $$640 \times 480 \times 24 \text{ bits} = 7.372.800 \text{ bits} = 921.600 \text{ bytes}$$

Όσο μεγαλύτερο είναι το πλαίσιο που θέλουμε να απεικονίσουμε, τόσο περισσότερη πρέπει να είναι και η μνήμη του υποσυστήματος γραφικών. Τα ηλεκτρονικά του υποσυστήματος γραφικών επιπλέον καθορίζουν την κάθετη συχνότητα ανανέωσης των πλαισίων, καθώς και το πλήθος των διαφορετικών χρωμάτων σε κάθε ανάλυση.

> Ένα υποσύστημα γραφικών με μνήμη 1 MB μπορεί να απεικονίσει σε μια οθόνη ανάλυση 640×480 με 24 bits για κάθε εικονοστοιχείο, αλλά για ανάλυση 800×600 μπορεί να παραστήσει το χρώμα κάθε εικονοστοιχείου μόνο με 16 bits (δηλαδή διαθέτει 2^{16} διαφορετικά χρώματα) και για ανάλυση 1024×768 διαθέτει μόνο 256 διαφορετικά χρώματα.

Η τεχνολογία κατασκευής και οι διαστάσεις της οθόνης περιορίζουν τη μέγιστη ανάλυση, το χρώμα και τη συχνότητα ανανέωσής της. Από εκεί και πέρα, το υποσύστημα γραφικών αναλαμβάνει να εκμεταλλευτεί τις δυνατότητες της οθόνης,

χρησιμοποιώντας αναλύσεις με τιμές μικρότερες ή ίσες των μεγίστων τιμών της οθόνης. Και αυτές οι αναλύσεις είναι τυποποιημένες και εξαρτώνται και από τις δυνατότητες του υποσυστήματος γραφικών (μνήμη, ηλεκτρονικά υποστήριξης κλπ).

Υπάρχουν όμως και άλλοι περιορισμοί όσον αφορά τις δυνατότητες απεικόνισης μιας οθόνης. Η μέγιστη συχνότητα οριζόντιας σάρωσης είναι δεδομένη· επομένως όσο μεγαλύτερη είναι η ανάλυση της εικόνας, τόσο μικρότερη είναι η συχνότητα κάθετης ανανέωσης. Σε παλαιότερες οθόνες η μέγιστη συχνότητα οριζόντιας σάρωσης ήταν μικρή, οπότε οι μεγάλες αναλύσεις απεικονίζονταν με πολύ μικρή συχνότητα σάρωσης. Το φαινόμενο αυτό ήταν ιδιαίτερα ενοχλητικό και κουραστικό για τα μάτια.

Μία λύση αποτέλεσε η *πεπλεγμένη* ή *πλεκτή σάρωση* (interlaced scanning). Στην πεπλεγμένη σάρωση διαιρούμε το ορατό πλαίσιο της εικόνας σε άρτιες και περιττές γραμμές. Κάθε φορά δεν ανανεώνουμε όλες τις γραμμές, αλλά εναλλάξ τις άρτιες και τις περιττές.

> Αν η κάθετη συχνότητα ανανέωσης της οθόνης μας είναι μόνο 30 Hz (δηλαδή τριάντα φορές το δευτερόλεπτο), η συχνότητα ανανέωσης των περιττών γραμμών είναι 30 φορές το δευτερόλεπτο, ενώ το ίδιο ισχύει και για τις άρτιες. Στην οθόνη απεικονίζονται εναλλάξ οι άρτιες και οι περιττές γραμμές δίνοντας ένα σύνολο 60 ορατών πλαισίων το δευτερόλεπτο που εναλλάσσονται. Το αποτέλεσμα της εναλλαγής αυτής των περιττών και άρτιων πλαισίων είναι να νομίζει ο χρήστης ότι έχει ανανέωση 60 πλαισίων το δευτερόλεπτο.

Η συνύπαρξη των περιττών γραμμών του τρέχοντος πλαισίου με τις άρτιες του προηγούμενου πλαισίου και αντίστροφα δημιουργεί *τρεμόσβησμα* (flickering) της οθόνης που είναι κουραστικό στην πολύωρη εργασία.

Οθόνη Υγρών Κρυστάλλων

Οι *οθόνες υγρών κρυστάλλων* (Liquid Crystal Display - LCD) βασίζονται στο φαινόμενο της πόλωσης της φωτεινής ακτινοβολίας όταν αυτή διαπερνά κάποιο υλικό σε κρυσταλλική μορφή.

Οι πρώτες τέτοιες οθόνες εμφανίστηκαν σε φορητές συσκευές, όπως είναι οι αριθμομηχανές τσέπης, τα ηλεκτρονικά ρολόγια, οι οθόνες ιατρικών μηχανημάτων κλπ. Στην αρχή οι οθόνες ήταν μονόχρωμες και μικρές σε μέγεθος, ενώ η ευκρίνεια απεικόνισής τους ήταν περιορισμένη. Στις μέρες μας υπάρχουν οθόνες 14 και 15 ιντσών με δυνατότητα απεικόνισης εκατομμυρίων χρωμάτων και ευκρίνεια αντίστοιχη των οθονών καθοδικού σωλήνα.

Στις οθόνες υγρών κρυστάλλων δεν υπάρχει καθοδικός σωλήνας, ούτε ηλεκτρονικά πυροβόλα. Η λειτουργία τους στηρίζεται στην ύπαρξη ενός πλέγματος από κύτταρα υγρών κρυστάλλων που είναι οργανωμένα σε γραμμές και στήλες. Τα κύτταρα κρυστάλλων ελέγχονται από τρανζίστορ. Όταν διοχετεύεται ρεύμα στα τρανζίστορ, αυτά περιστρέφουν τους κρυστάλλους κατά γωνία ανάλογη της έντασης του ρεύματος.

Στο πίσω μέρος της οθόνης υπάρχει μια φθοριούχος πλάκα που ακτινοβολεί φως, το οποίο διέρχεται μέσα από δύο φίλτρα οριζόντιας και κατακόρυφης πόλωσης, στη μέση των οποίων υπάρχει το πλέγμα των κρυστάλλων και το φίλτρο των χρωμάτων. Τα φίλτρα οριζόντιας και κατακόρυφης πόλωσης αφήνουν να περάσει ακτινοβολία συγκεκριμένου μήκους κύματος με οριζόντια ή κατακόρυφη ταλάντωση. Στη συνέχεια το πλέγμα των κρυστάλλων στρέφει την ακτίνα του φωτός ώστε να περάσει ολόκληρη, λίγη ή καθόλου μέσα από το επόμενο φίλτρο, με αποτέλεσμα στην οθόνη να εμφανίζεται το τελικό ίχνος.

Οι κυριότερες κατηγορίες οθονών υγρών κρυστάλλων είναι οι *οθόνες ενεργής μήτρας* (active matrix) και οι *οθόνες παθητικής μήτρας* (passive matrix). Στις ενεργές οθόνες, κάθε κύτταρο ελέγχεται από ένα ξεχωριστό τρανζίστορ. Οι οθόνες αυτές έχουν καλύτερη ευκρίνεια άλλα είναι πολύ πιο ακριβές.

Λόγω της απουσίας του καθοδικού σωλήνα, οι οθόνες υγρών κρυστάλλων είναι ελαφρύτερες και λιγότερο ογκώδεις από τις οθόνες καθοδικού σωλήνα, με αποτέλεσμα να αποτελούν ιδανική λύση για φορητούς υπολογιστές.

Το κόστος κατασκευής τους όμως είναι αρκετά μεγάλο. Η τεχνολογία που χρησιμοποιείται, εμφανίζει μεγάλα ποσοστά ελαττωματικών οθονών, και μόνο ένα μικρό ποσοστό όσων κατασκευάζονται, είναι τελικά αποδεκτό για πώληση. Όσο μεγαλώνει η διάμετρος της οθόνης τόσο πιο πιθανό είναι να εμφανιστεί λάθος κατά την κατασκευή της οθόνης, άρα αυξάνεται το κόστος κατασκευής.

Στις μέρες μας κυκλοφορούν οθόνες υγρών κρυστάλλων διαγωνίου μέχρι και 15 ιντσών, που χρησιμοποιούνται όχι μόνο σε φορητούς υπολογιστές αλλά και σε υπολογιστές γραφείου, καθώς εκπέμπουν πολύ λιγότερη ακτινοβολία και είναι πολύ μικρότερες σε όγκο.

Οθόνη Αερίου-Πλάσματος (Gas-Plasma Display)

Η κατηγορία αυτή των οθονών βασίζεται στην τεχνολογία των λαμπτήρων με αέριο νέον: το αέριο παράγει φωτεινή ακτινοβολία με την παρουσία ηλεκτρικού ρεύματος. Υπάρχουν ηλεκτρόδια πάνω και κάτω από ποσότητες αερίου, τα οποία διοχετεύουν επιλεκτικά ηλεκτρικό ρεύμα διεγείροντας το αέριο που φωτοβολεί. Η τεχνολογία αυτή είναι αρκετά υποσχόμενη και στο μέλλον θα εξαπλωθεί περισσότερο, ακόμα και σε υπολογιστές γραφείου.

Ανακεφαλαίωση

Η πιο συνηθισμένη κατηγορία οθονών υπολογιστών είναι οι οθόνες καθοδικού σωλήνα. Η λειτουργία τους στηρίζεται στην ύπαρξη ενός πυροβόλου που εκτοξεύει ηλεκτρόνια σε φωσφορίζουσα επιφάνεια προκαλώντας την ακτινοβολία της. Τα κυριότερα χαρακτηριστικά των οθονών καθοδικού σωλήνα είναι η ανάλυση, η συχνότητα οριζόντιας σάρωσης και η συχνότητα ανανέωσης πλαισίου.

Οι κυριότερες κατηγορίες οθονών υγρών κρυστάλλων είναι οι οθόνες ενεργής μήτρας και οι οθόνες παθητικής μήτρας. Οι οθόνες αυτές βασίζονται στην τεχνολογία της πόλωσης του φωτός όταν διέρχεται από κρυσταλλική δομή. Με την παροχή ρεύματος μέσω τρανζίστορ, προκαλείται επιλεκτικά περιστροφή του κρυστάλλου και πόλωση του φωτός που διέρχεται από μέσα. Η τεχνολογία των υγρών κρυστάλλων προσφέρει οθόνες λεπτότερες, μικρότερες σε όγκο και ελαφρύτερες, όμως ειδικά σε φθηνότερες οθόνες υπάρχουν προβλήματα απεικόνισης. Στο μέλλον αναμένεται να επικρατήσουν ακόμα και σε επιτραπέζιους υπολογιστές γραφείου.

Γλωσσάριο όρων

Ανάλυση	*Resolution*
Βήμα Κουκίδας	*Dot Pitch*
Εικονοστοιχείο	*Pixel - Picture Element*
Κάθοδος	*Cathode*
Κατακόρυφη Συχνότητα	*Vertical Scanning Frequency*
Μάσκα	*Mask*
Οθόνη	*Screen*
Οθόνη Αερίου-Πλάσματος	*Gas-Plasma Display*
Οθόνη Ενεργής Μήτρας	*Active Matrix*
Οθόνη Καθοδικού Σωλήνα	*Cathode Ray Tube*
Οθόνη Παθητικής Μήτρας	*Passive Matrix*
Οθόνη Υγρών Κρυστάλλων	*Liquid Crystal Display*
Οριζόντια Συχνότητα Σάρωσης	*Horizontal Scanning Frequency*
Πλεκτή Σάρωση	*Interlaced Scanning*
Συχνότητα Ανανέωσης Πλαισίου	*Refresh Rate*
Τρεμόσβησμα	*Flickering*
Φωτεινότητα	*Brightness*
Χρόνος Αναλαμπής	*Persistence*

Ερωτήσεις

? Πώς λειτουργούν οι μονόχρωμες οθόνες με καθοδικό σωλήνα;

? Πώς λειτουργούν οι οθόνες με αποχρώσεις του γκρί με καθοδικό σωλήνα;

? Πώς λειτουργούν οι έγχρωμες οθόνες με καθοδικό σωλήνα;

? Πώς λειτουργούν οι έγχρωμες οθόνες υγρών κρυστάλλων;

Μάθημα 5.4 Μαγνητικά Μέσα Αποθήκευσης

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι να περιγράψει τον τρόπο λειτουργίας, τις κατηγορίες και τα χαρακτηριστικά των μαγνητικών μέσων αποθήκευσης στους σύγχρονους υπολογιστές.

Τι θα μάθεις;

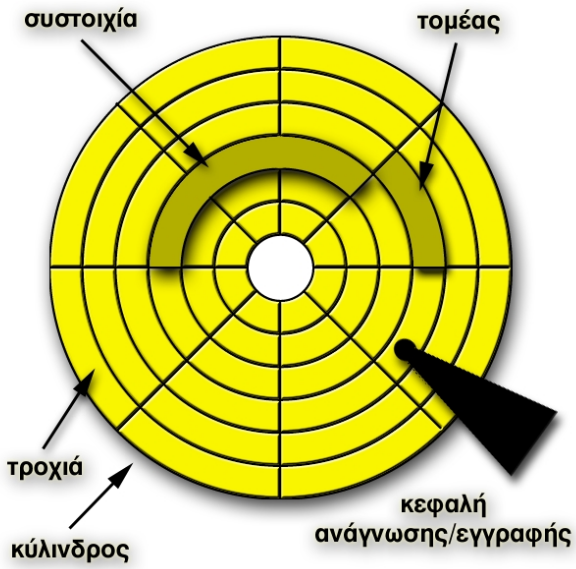
Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τη δομή και την οργάνωση των μονάδων μαγνητικών δίσκων*
- *Να περιγράφεις τη γεωμετρία ενός δίσκου*
- *Να διακρίνεις τις κατηγορίες μονάδων μαγνητικών ταινιών*

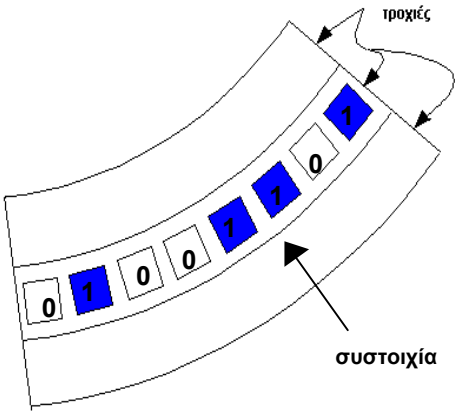
Τα μαγνητικά μέσα αποθήκευσης είναι οι *ταινίες* (magnetic tapes) και οι *δίσκοι* (disks). Η εγγραφή των ψηφιακών δεδομένων σε αυτά βασίζεται στη μόνιμη μαγνήτιση μιας μικρής περιοχής του μαγνητικού μέσου με την βοήθεια ενός ηλεκτρομαγνήτη. Η ανάγνωση γίνεται με το πέρασμα του ηλεκτρομαγνήτη πάνω από τη μαγνητισμένη περιοχή και τη δημιουργία επαγωγικού ρεύματος. Η φορά του επαγωγικού ρεύματος εξαρτάται από την πολικότητα της μαγνήτισης και επομένως καθορίζει τη τιμή της ψηφιακής πληροφορίας: 0 ή 1. Στην περίπτωση της εγγραφής έχουμε το αντίστροφο φαινόμενο. Ανάλογα με την τιμή 0 ή 1 που θέλουμε να γραφεί στο δίσκο, διοχετεύεται ρεύμα αντίστοιχης φοράς, δημιουργείται λοιπόν μαγνητικό πεδίο που μαγνητίζει αντίστοιχα την περιοχή του δίσκου.

Οργάνωση μαγνητικών δίσκων

Η επιφάνεια ενός μαγνητικού δίσκου διαμερίζεται σε ομόκεντρους κύκλους ή *τροχιές* (tracks). Κάθε τροχιά χωρίζεται σε κυκλικά τόξα ή περιοχές που ονομάζονται *τομείς* (sectors). Κάθε τομέας έχει μέγεθος σταθερό, με άλλα λόγια περιέχει σταθερό αριθμό από bytes, ο οποίος κυμαίνεται από 512 bytes ως 32 KB ή και περισσότερο. Το σύνολο των δεδομένων που αποθηκεύονται σε ένα τομέα του δίσκου λέγεται *ενότητα* (block). Δύο ή περισσότεροι τομείς, σχηματίζουν την *συστοιχία* (cluster), που αποτελεί την μικρότερη μονάδα αποθήκευσης για τα δεδομένα ενός προγράμματος.

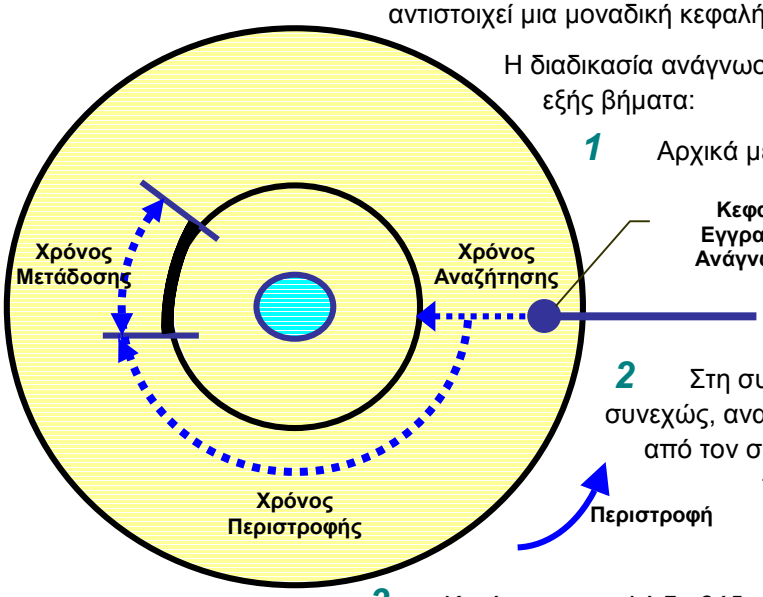


Όπως μπορούμε να δούμε και στο σχήμα, το μέγεθος των τομέων που ανήκουν σε διαφορετικές τροχιές δεν είναι το ίδιο. Οι τομείς που βρίσκονται κοντά στην εξωτερική τροχιά του δίσκου έχουν μεγαλύτερο μήκος απ' ό,τι οι εσωτερικοί τομείς. Η ποσότητα όμως των δεδομένων που αποθηκεύεται σε κάθε τομέα είναι συνήθως σταθερή· άρα έχουμε μεγαλύτερη πυκνότητα αποθήκευσης όσο πλησιάζουμε προς το κέντρο του δίσκου.



Πάνω από την επιφάνεια του δίσκου και σε μικρή απόσταση από αυτόν (εκατομυριοστά του μέτρου) βρίσκεται η *κεφαλή* (head) ανάγνωσης/εγγραφής, η οποία φέρει τον ηλεκτρομαγνήτη για την ανάγνωση των 0 και 1 που είναι γραμμένα στην επιφάνεια του δίσκου. Οι κεφαλές μπορεί να είναι σταθερές ή κινούμενες. Στην περίπτωση των σταθερών κεφαλών, πάνω από την επιφάνεια του μαγνητικού δίσκου βρίσκονται πολλές κεφαλές, μια για κάθε τροχιά. Οι κινούμενες κεφαλές είναι οι πλέον διαδεδομένες σε μονάδες μαγνητικών δίσκων. Όταν οι κεφαλές είναι κινούμενες, σε κάθε επιφάνεια του μαγνητικού δίσκου αντιστοιχεί μια μοναδική κεφαλή.

Η διαδικασία ανάγνωσης/εγγραφής δεδομένων αποτελείται από τα εξής βήματα:



1 Αρχικά μετακινείται η κεφαλή πάνω από την τροχιά στην οποία υπάρχουν ή πρόκειται να γραφούν τα δεδομένα. Ο χρόνος που απαιτείται για τη μετακίνηση αυτή ονομάζεται *χρόνος αναζήτησης* (seek time).

2 Στη συνέχεια, καθώς ο δίσκος περιστρέφεται συνεχώς, αναμένουμε να βρεθεί η κεφαλή ακριβώς πάνω από τον συγκεκριμένο τομέα και συστοιχία. Ο χρόνος που απαιτείται μέχρι να γίνει αυτό εξαρτάται από το *ρυθμό περιστροφής* του μαγνητικού δίσκου.

3 Κατόπιν η κεφαλή διαβάζει ή γράφει τα δεδομένα σε συστοιχίες και τα μεταδίδει. Ο *χρόνος ανάγνωσης* ή *εγγραφής* εξαρτάται από την ταχύτητα ανάγνωσης/εγγραφής της κεφαλής, από την ταχύτητα των ηλεκτρονικών που υποστηρίζουν το δίσκο, από το πρωτόκολλο διαδρόμου κλπ.

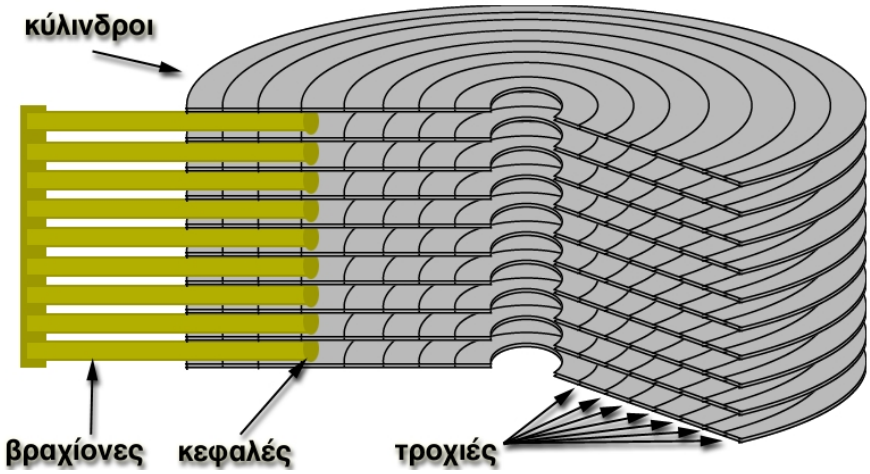
Οι μαγνητικοί δίσκοι διακρίνονται σε *σκληρούς δίσκους* (hard disks) και σε εύκαμπτες *δισκέτες* (floppy disks).

Σκληροί δίσκοι

Οι σκληροί δίσκοι αποτελούν το κυριότερο μέσο αποθήκευσης στους σύγχρονους υπολογιστές. Αποτελούνται από δύο η περισσότερους ομοαξονικούς μαγνητικούς δίσκους που περιστρέφονται συνεχώς με την ίδια γωνιακή ταχύτητα σε μικρή απόσταση μεταξύ τους. Οι δίσκοι κατασκευάζονται από μεταλλικό ή κεραμικό κράμα, κατά προτίμηση ελαφρύ αλλά ανθεκτικό σε υψηλές θερμοκρασίες. Η χρήση υλικού ανθεκτικού στη θερμότητα είναι απαραίτητη, καθώς οι σκληροί δίσκοι περιστρέφονται με χιλιάδες στροφές ανά λεπτό, σχεδόν συνεχώς, και επομένως αναπτύσσονται

μεγάλες θερμοκρασίες στο εσωτερικό τους. Στις εξωτερικές επιφάνειες του δίσκου υπάρχει λεία επίστρωση από μαγνητικό υλικό.

Οι κεφαλές ανάγνωσης είναι συνήθως για καθένα δίσκο, μια σε κάθε πλευρά. Για να επιτευχθούν μεγάλες ταχύτητες ανάγνωσης/εγγραφής δεδομένων, οι κεφαλές βρίσκονται σε πολύ μικρή απόσταση από την επιφάνεια. Οι σκληροί δίσκοι είναι συσκευασμένοι σε κενό αέρα, καθώς το παραμικρό μόριο σκόνης ή άλλης ακαθαρσίας μπορεί να καταστρέψει τον δίσκο.



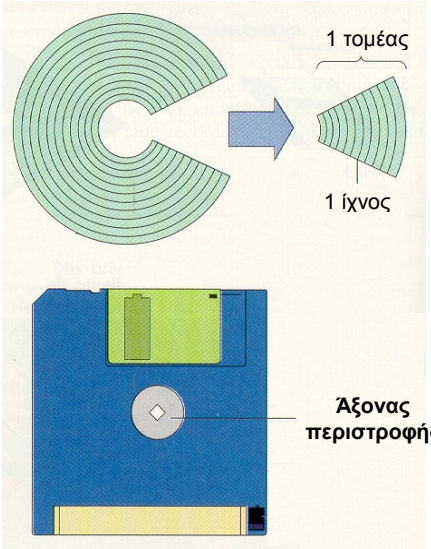
Στο σχήμα βλέπουμε μια τυπική διάταξη σκληρού δίσκου που αποτελείται από ομοαξονικούς μαγνητικούς δίσκους. Οι αντίστοιχες τροχιές από κάθε δίσκο, ορίζουν ένα *κύλινδρο* (cylinder). Σε κάθε στιγμή, όλες οι κεφαλές είναι τοποθετημένες στις τροχιές που ανήκουν στον ίδιο κύλινδρο. Έτσι η διαδικασία προσπέλασης γίνεται ταχύτερα, καθώς όλες οι κεφαλές λειτουργούν ταυτόχρονα γράφοντας στην ίδια τροχιά, αλλά σε διαφορετικό μαγνητικό δίσκο. Τα διαδοχικά δεδομένα λοιπόν αποθηκεύονται σε τροχιές διαφορετικών μαγνητικών δίσκων που ανήκουν στον ίδιο κύλινδρο.

Για την προσπέλαση των δεδομένων στο σκληρό δίσκο, υπάρχει μια μοναδική διεύθυνση για κάθε τομέα, η οποία αποτελείται από τον αριθμό της κεφαλής (δηλαδή τον αριθμό δίσκου), τον αριθμό του κυλίνδρου ή τροχιάς στο δίσκο και τον αριθμό του τομέα μέσα στον κύλινδρο. Η χωρητικότητα (capacity) σε Kbytes (KB) ενός σκληρού δίσκου δίνεται από το γινόμενο:

σύνολο κεφαλών × αριθμός κυλίνδρων × αριθμός τομέων/κύλινδρο × KB/τομέα

Τα χαρακτηριστικά αυτά (κεφαλές, κύλινδροι, τομείς) αποτελούν τη *γεωμετρία* του σκληρού δίσκου**.** Η σύγχρονη τεχνολογία επιτρέπει την δημιουργία δίσκων με πυκνότερες τροχιές και πιο μικρούς τομείς ανά μονάδα πληροφορίας, άρα μεγαλύτερες χωρητικότητες.

> Ένας δίσκος με 8 κεφαλές, 256 κυλίνδρους και 128 τομείς ανά κύλινδρο έχει 8×256×128=262.144 τομείς και αν ο τομέας έχει 512 bytes, τότε η χωρητικότητα του είναι 134.217.728 bytes ή 128 MB.

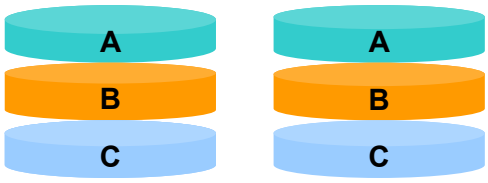


Δισκέτες

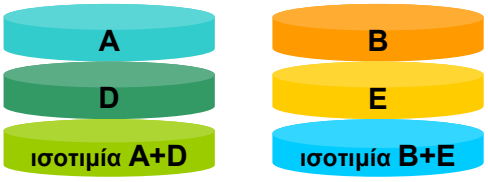
Οι **δισκέτες** αποτελούνται από έναν εύκαμπτο πλαστικό δίσκο πάνω στο οποίο βρίσκεται το μαγνητικό υλικό, ενώ το εξωτερικό τους περίβλημα αποτελείται από πλαστικό, προστατεύοντάς τις από υγρασία και σκόνη. Η εγγραφή πάνω στον εύκαμπτο μαγνητικό δίσκο γίνεται και από τις δύο πλευρές, δίνοντας συνολική χωρητικότητα 1,44MB για διάμετρο δίσκου 3,5 ιντσών, και 1,2 MB για διάμετρο 5 ¼ ιντσών. Στις δισκέτες η κεφαλή ανάγνωσης/εγγραφής δε βρίσκεται σε απόσταση από την επιφάνεια του μαγνητικού δίσκου, αλλά έρχεται σε άμεση επαφή με αυτόν.

Επιπλέον, οι δισκέτες περιστρέφονται μόνο όταν πρόκειται να γίνει προσπέλαση δεδομένων και όχι συνεχώς. Εξαιτίας των παραπάνω, η ταχύτητα περιστροφής μιας δισκέτας είναι πολύ μικρότερη από αυτή του σκληρού δίσκου, γύρω στις 300 στροφές/λεπτό, και η ταχύτητα προσπέλασης δεδομένων πολύ μικρότερη.

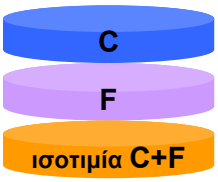
Μονάδες Δίσκων RAID



Οι *μονάδες RAID* (Redundant Array of Inexpensive/Independent Disks) είναι συστοιχίες δύο ή περισσοτέρων κοινών σκληρών δίσκων προσφέροντας κυρίως μεγαλύτερη απόδοση και ανοχή σε σφάλματα. Η απόδοση των δίσκων αυξάνεται, καθώς τα δεδομένα δεν αποθηκεύονται σειριακά στους δίσκους αλλά παράλληλα, όπως υπαγορεύει το πρότυπο του RAID. Έτσι διαδοχικά bytes ενός τομέα ή ακόμα και τα bits του ίδιου byte γράφονται σε ξεχωριστούς δίσκους, ώστε η εγγραφή και η ανάγνωση να γίνονται με παράλληλη λειτουργία όλων των δίσκων. Ένας ελεγκτής RAID αναλαμβάνει το συνδυασμό των bits ή των bytes, ανάλογα με την υλοποίηση, ώστε ο υπολογιστής να βλέπει ένα «ενιαίο» σκληρό δίσκο. Η ανοχή σε σφάλματα επιτυγχάνεται με δύο τρόπους:



- Με *καθρεπτισμό* των δίσκων (mirroring)
- Με την ύπαρξη δίσκου με *bit ισοτιμίας* (parity bit).



Στον καθρεπτισμό υπάρχει ένα πανομοιότυπο αντίγραφο κάθε δίσκου, ώστε σε περίπτωση απώλειας του πρωτότυπου να υπάρχει το αντίγραφο. Στην ισοτιμία, ένας δίσκος αποθηκεύει το bit ισοτιμίας άλλων δύο δίσκων, ώστε σε περίπτωση απώλειας του ενός από αυτούς, με τον δίσκο ισοτιμίας και τον άλλο να μπορούμε να ανακτήσουμε τα δεδομένα αυτού που χάθηκε. Προφανώς και οι δύο λύσεις μας επιβαρύνουν με ύπαρξη δίσκων εφεδρείας που λειτουργούν ως καθρέπτες ή ως δίσκοι ισοτιμίας. Το κόστος απώλειας των δεδομένων όμως είναι πάντα πολύ μεγαλύτερο από αυτό των επιπλέον μονάδων δίσκων.

Μονάδες Εφεδρικής Αποθήκευσης

Οι *μονάδες εφεδρείας* (backup units) χρησιμοποιούνται για τη δημιουργία αντιγράφων ασφαλείας των δεδομένων που είναι αποθηκευμένα στους υπολογιστές. Στην κατηγορία αυτή ανήκουν κυρίως οι μαγνητικές ταινίες, οι μαγνητοοπτικοί δίσκοι, ενώ τελευταία χρησιμοποιούνται και οι οπτικοί δίσκοι CD-RW και DVD. Οι οπτικοί δίσκοι θα αναλυθούν στο επόμενο μάθημα.

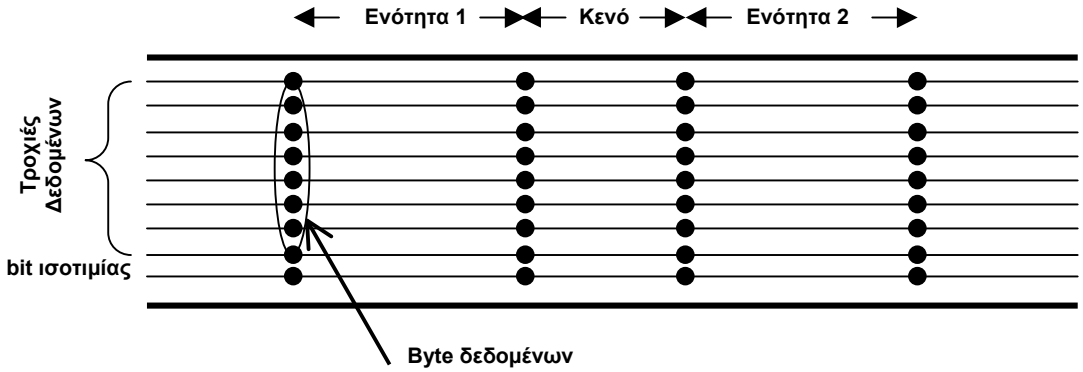
Μαγνητοοπτικοί δίσκοι

Οι μαγνητοοπτικοί δίσκοι έχουν τις ίδιες διαστάσεις με μια εύκαμπτη δισκέτα 5 ¼ ιντσών, αλλά είναι παχύτεροι. Η χωρητικότητα τους ποικίλλει από 128 MB έως 1,2 GB. Συνδυάζουν τη μαγνητική τεχνολογία κατά την εγγραφή τους και την οπτική κατά την ανάγνωση. Έχουν επομένως μεγαλύτερη διάρκεια ζωής από τις μαγνητικές ταινίες, καθώς οι πολλαπλές αναγνώσεις δεν αλλοιώνουν τα αποθηκευμένα δεδομένα. Από την άλλη πλευρά είναι σημαντικά ακριβότερα μέσα αποθήκευσης, ενώ σημαντικά κοστίζουν και οι περιφερειακές συσκευές που τους υποστηρίζουν.

Τελευταία έχουν εμφανιστεί οδηγοί ανάγνωσης δισκετών, οι οποίοι διαβάζουν και εύκαμπτους οπτικούς δίσκους, δισκέτες δηλαδή που έχουν την ίδια μαγνητοοπτική τεχνολογία και χωρητικότητα που κυμαίνεται από 21-120 MB δεδομένων.

Μαγνητικές ταινίες

Στις μονάδες *μαγνητικών ταινιών* (magnetic tapes) οι πληροφορίες καταγράφονται σε μια εύκαμπτη πλαστική ταινία, στην επιφάνεια της οποίας υπάρχει σιδηρομαγνητικό υλικό. Οι πληροφορίες αποθηκεύονται συνήθως σε 9 (ή πιο σπάνια σε 7) παράλληλες διαμήκεις τροχιές, τα *ίχνη* (tracks) στη μια επιφάνεια της ταινίας. Η μαγνητική κεφαλή μπορεί να διαβάζει ή να γράφει παράλληλα και τα εννέα ίχνη, επιτρέποντας έτσι την



ταυτόχρονη ανάγνωση ή εγγραφή δεδομένων μεγέθους 9 bits. Συχνά οι 8 τροχιές χρησιμοποιούνται για την αποθήκευση ενός byte ενώ το 9^{ο} bit χρησιμοποιείται για την αποθήκευση του *bit ισοτιμίας* (parity bit), ώστε να ανιχνεύονται πιθανά λάθη ανάγνωσης.

Όταν πρόκειται να γίνει ανάγνωση δεδομένων από την ταινία, τα ηλεκτρονικά κυκλώματα υποστήριξης της μονάδας (δηλαδή ο ελεγκτής ταινίας) φροντίζουν για την περιστροφή της στο κατάλληλο σημείο. Η προσπέλαση των δεδομένων γίνεται ακολουθιακά. Η ταχύτητα περιστροφής της ταινίας κατά την ανάγνωση σε συνδυασμό με την πυκνότητα εγγραφής των δεδομένων στην επιφάνεια της, καθορίζουν την *ταχύτητα μετάδοσης* (transmission rate) δεδομένων από την ταινία.

Οι κυριότερες κατηγορίες ταινιών είναι:

9-Track ή **ταινία μισής ίντσας**: Παρόλο που αυτοί οι τύποι ταινιών εξακολουθούν να χρησιμοποιούνται, είναι πολύ μικρών δυνατοτήτων αποθήκευσης με βάση τα σημερινά δεδομένα. Οι ταινίες αυτές κατά τη μέγιστη απόδοση τους μπορούν να αποθηκεύσουν μέχρι 225MB.

QIC ή **ταινία ενός τετάρτου**: Η δυνατότητα αποθήκευσης αυτών των ταινιών έφθανε τα 2 GB, ωστόσο συνήθως οι ταινίες αυτές είχαν δυνατότητες αποθήκευσης 150 MB, 320 MB και 525 MB.

DAT (Digital Audio Tape): Οι *ψηφιακές ταινίες* εμφανίζονται σε δύο σταθερά πλάτη: 8 mm και 4 mm. Αυτή των 4mm καλύπτει ανάγκες αποθήκευσης από 1 έως 4 GB (16 GB με συμπίεση) ενώ η ταινία των 8 mm καλύπτει ανάγκες από 2 έως 10 GB. Οι ταινίες των 8 mm και 4 mm είναι οι πιο συνηθισμένες στα νέα υπολογιστικά συστήματα. Η ταινία πλάτους 4 mm είναι φυσικά η μικρότερη σε μέγεθος μαγνητική ταινία, γι' αυτό έχει και τη μικρότερη αποθηκευτική δυνατότητα. Το μόνο σημαντικό μειονέκτημα αυτών των ταινιών είναι ότι είναι περισσότερο ευαίσθητες σε θερμοκρασιακές καταπονήσεις απ' ό,τι άλλα είδη ταινιών. Οι ταινίες των 4 mm ήταν οι πιο ευρέως χρησιμοποιούμενες όμως έχουν αρχίσει να αντικαθίστανται από τις DLT.

DLT (Digital Linear Tape): Οι *ψηφιακές γραμμικές ταινίες* έχουν αποθηκευτική δυνατότητα έως και 40 GB. Οι συσκευές που τις χειρίζονται είναι αρκετά γρήγορες και αποτελούν τη νέα μέθοδο αποθήκευσης στην τεχνολογία των πολυμέσων.

Τα τελευταία χρόνια χρησιμοποιούνται για εφεδρεία και οπτικά μέσα, τα οποία αναλύονται στο επόμενο μάθημα.

Ανακεφαλαίωση

Οι μαγνητικοί δίσκοι αποτελούν το κύριο μέσο αποθήκευσης στους υπολογιστές. Η εγγραφή τους βασίζεται στη μαγνήτιση της επιφάνειας του δίσκου και η ανάγνωση στη δημιουργία επαγωγικού ρεύματος ορθής ή ανάστροφης φοράς (που αντιστοιχεί σε 0 ή 1) κατά το πέρασμα ενός ηλεκτρομαγνήτη (της κεφαλής) πάνω από τη μαγνητισμένη επιφάνεια. Οι σκληροί δίσκοι αποτελούνται από έναν ή περισσότερους ομόκεντρους μαγνητικούς δίσκους. Οι δίσκοι διαιρούνται σε τροχιές, τομείς και συστοιχίες.

Οι μονάδες εφεδρείας βασίζονται κυρίως σε μαγνητικά μέσα. Έτσι υπάρχουν οι μαγνητοοπτικοί δίσκοι και οι ταινίες διαφόρων διαμετρημάτων και χωρητικοτήτων.

Γλωσσάριο όρων

Bit Ισοτιμίας	*Parity Bit*
Ενότητα	*Block*
Εύκαμπτη Δισκέτα	*Floppy Disk*
Καθρεπτισμός Των Δίσκων	*Mirroring*
Κεφαλή Ανάγνωσης/Εγγραφής	*Read/Write Head*
Κύλινδρος	*Cylinder*
Μαγνητικές Ταινίες	*Magnetic Tapes*
Μαγνητικοί Δίσκοι	*Magnetic Disks*
Μονάδες Δίσκων RAID	*Redundant Array of Inexpensive/Independent Disks*
Μονάδες Εφεδρείας	*Backup Units*
Σκληρός Δίσκος	*Hard Disk*
Συστοιχία	*Cluster*
Ταχύτητα Μετάδοσης	*Transmission Rate*
Τομείς	*Sectors*
Τροχιές	*Tracks*
Χρόνος Αναζήτησης	*Seek Time*
Χωρητικότητα	*Capacity*

Ερωτήσεις

? Πώς είναι οργανωμένος ένας σκληρός δίσκος;

? Περίγραψε μία μονάδα σκληρού δίσκου.

? Περίγραψε μία δισκέτα.

? Τι είναι οι μονάδες εφεδρείας;

Μάθημα 5.5 Οπτικά Μέσα Αποθήκευσης

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι να περιγράψει τον τρόπο λειτουργίας, τις κατηγορίες και τα χαρακτηριστικά των οπτικών μέσων αποθήκευσης στους σύγχρονους υπολογιστές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

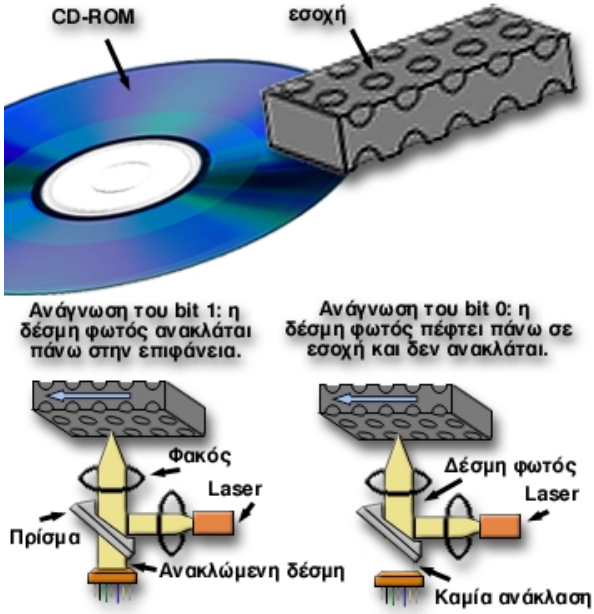
- *Να περιγράφεις την τεχνολογία ανάγνωσης στους δίσκους CD*
- *Να απαριθμείς τις κατηγορίες των οπτικών δίσκων*
- *Να απαριθμείς τις τεχνολογίες εγγραφής πληροφοριών στους οπτικούς δίσκους*
- *Να εξηγείς τί είναι δίσκοι CD-Audio, εγγράψιμοι/επανεγγράψιμοι δίσκοι, δίσκοι DVD*

Οπτικοί Δίσκοι

Οι *οπτικοί δίσκοι* (Compact Disk) είναι τα τελευταία χρόνια το πιο διαδεδομένο μέσο αποθήκευσης μεγάλου όγκου ψηφιακών δεδομένων, ειδικά για εφαρμογές πολυμέσων. Οι συσκευές ανάγνωσης οπτικών δίσκων βασίζονται στην ανάκλαση μιας φωτεινής δέσμης laser πάνω στην επιφάνεια του δίσκου. Το υλικό κατασκευής τους ποικίλλει ανάλογα με την τεχνολογία ανάγνωσης και εγγραφής. Το κοινό χαρακτηριστικό τους είναι η μεγάλη διάρκεια ζωής τους λόγω του προστατευτικού στρώματος πλαστικού που τους περιβάλλει. Οι πιο διαδεδομένες κατηγορίες οπτικών δίσκων είναι οι εξής:

- ✓ **CD-ROM**: Μουσικά CD (CD-Audio) και CD δεδομένων (CD-Data)
- ✓ **Εγγράψιμα CD** (CD-Recordable ή WORM-Write Once Read Many)
- ✓ **Επανεγγράψιμα CD** (CD-RW Rewritable ή CD-Erasable)
- ✓ **DVD** (Digital Versatile Disks)

Χάρη στο οπτικό σύστημα σάρωσης, δεν υπάρχει τριβή ανάμεσα στο σύστημα ανάγνωσης και το δίσκο, με αποτέλεσμα να μη φθείρεται ο δίσκος, όσο συχνά κι αν χρησιμοποιείται. Πρέπει ωστόσο να είμαστε προσεκτικοί στη χρήση των οπτικών δίσκων, καθώς γρατσουνιές και σκόνη μπορούν να διακόψουν ή να εμποδίσουν τη δέσμη φωτός με αποτέλεσμα κάποιο τμήμα των δεδομένων να υπερπηδηθεί ή και να χαθεί. Τέτοια προβλήματα μπορούν να λυθούν αν κατά την εγγραφή προστεθεί ένας «κώδικας διόρθωσης λαθών» (CIRC) που αυτόματα αναπληρώνει κάθε χαμένη ή αλλοιωμένη πληροφορία κάνοντας μια σειρά από μαθηματικούς υπολογισμούς. Χωρίς αυτόν τον κώδικα διόρθωσης λαθών δεν θα μπορούσε να λειτουργήσει αξιόπιστα ο οπτικός δίσκος καθώς και η παραμικρή δόνηση του δαπέδου θα μπορούσε να επιφέρει ακουστικές ή οπτικές διαταραχές.

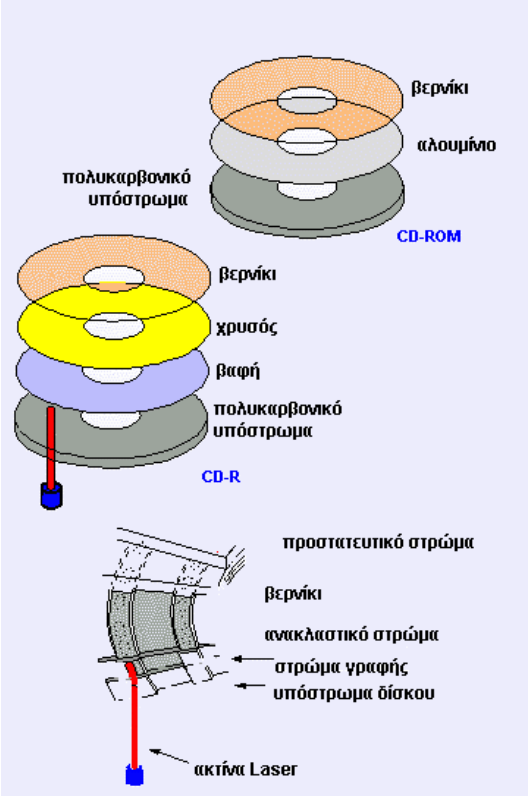


Για να είναι ο ρυθμός ανάγνωσης δεδομένων σταθερός, ο δίσκος στρέφεται με μεταβλητή γωνιακή ταχύτητα (ρυθμός περιστροφής) : μικρότερη αν η δέσμη laser προσπίπει κοντά στο κέντρο του δίσκου και μεγαλύτερη όταν προσπίπει κοντά στην περιφέρεια του δίσκου. Τα δεδομένα είναι γραμμένα σε σπειροειδείς τροχιές· έτσι, όσο πιο κοντά βρισκόμαστε στο κέντρο του δίσκου τόσο πιο μικρό είναι το μήκος της τροχιάς και επομένως τόσο λιγότερα τα αποθηκευμένα δεδομένα. Συνεπώς ο ρυθμός περιστροφής όταν γίνεται ανάγνωση κοντά στο κέντρο του δίσκου είναι μικρός. Όταν η δέσμη laser προσπίπει σε μία τροχιά κοντά στην περιφέρεια, ο ρυθμός περιστροφής είναι μεγαλύτερος. Έτσι ο ρυθμός ανάγνωσης είναι σε κάθε περίπτωση σταθερός. Η γωνιακή ταχύτητα περιστροφής κυμαίνεται από 268 ως 568 κύκλους το λεπτό, δίνοντας μια σταθερή γραμμική ταχύτητα 1,2-1,4 m/sec.

CD-ROM

Ο πιο διαδεδομένος τύπος οπτικών δίσκων που ξεκίνησε από το χώρο της ψηφιακής μουσικής είναι τα CD-ROM.

Ο δίσκος αυτός έχει διάμετρο 120 mm. Τα δεδομένα είναι γραμμένα πάνω σε ένα ειδικό στρώμα αλουμινίου, καλυμμένο από διαδοχικές προστατευτικές στρώσεις πλαστικού. Πάνω στο αλουμινένιο στρώμα σχηματίζονται *κοιλότητες* ή *εσοχές* (pits) και λείες επιφάνειες. Η δέσμη ανάγνωσης πέφτει πάνω στο αλουμινένιο στρώμα. Αν πέσει επάνω σε εσοχή, τότε διαθλάται, και η ισχύς της ακτινοβολίας που ανακλάται προς τα πίσω είναι πολύ μικρότερη από την προσπίπτουσα. Ο αισθητήρας της ανακλώμενης δέσμης αναγνωρίζει τη μείωση της ισχύος και αντιστοιχίζει σε αυτήν το 0. Αν η δέσμη ανάγνωσης είχε πέσει σε λεία επιφάνεια και είχε ανακλαστεί πλήρως, τότε θα αντιστοίχιζε σε αυτήν το 1. Επομένως οι εσοχές αντιστοιχούν στο 0 και οι λείες επιφάνειες στο 1.



Υπάρχουν 2 κατηγορίες CD ανάλογα με το αν περιέχουν μουσική (CD-DA - Digital Audio) ή ψηφιακά δεδομένα (CD-Data). Και στις δύο περιπτώσεις ο δίσκος διαιρείται σε 333.000 τομείς των 2352 bytes.

Στα **CD δεδομένων** τα 2048 bytes από κάθε τομέα χρησιμοποιούνται για την αποθήκευση δεδομένων και τα υπόλοιπα είναι για τον καθορισμό της αρχής ή του τέλους του τομέα, την αρίθμησή του, καθώς και κώδικες διόρθωσης λαθών. Η συνολική τους χωρητικότητα φτάνει τα 650 MB.

Στα **μουσικά CD**, ο ήχος είναι αποθηκευμένος σε παλμοκωδική διαμόρφωση (PCM), με συχνότητα δειγματοληψίας 44,1 KHz και 2 κανάλια ήχου, με δείγματα των 16 bits (δηλαδή αποθηκεύονται 176.400 bytes για κάθε δευτερόλεπτο μουσικής, που αντιστοιχούν σε 75 τομείς). Στα CD αυτά χρησιμοποιούνται όλα τα bytes του κάθε τομέα για την αποθήκευση του ήχου και δεν υπάρχουν αριθμήσεις τομέων κλπ. Τα τραγούδια γράφονται συνεχόμενα, με ένα μικρό διάκενο μεταξύ τους για να το ανιχνεύει η μονάδα ανάγνωσης. Είναι φανερό λοιπόν ότι η συνολική χωρητικότητα των CD που αποθηκεύουν ήχο είναι μεγαλύτερη

από τα 650 MB, καθώς χρησιμοποιούνται όλα τα bytes κάθε τομέα. Στα μουσικά CD ο συνολικός διαθέσιμος αποθηκευτικός χώρος είναι 74 λεπτά.

Εγγράψιμα CD (CD-R)

Η διαδικασία εγγραφής σε αυτή την κατηγορία οπτικών δίσκων έχει ουσιαστικές διαφορές από τα κοινά CD. Οι δίσκοι αυτοί αποτελούνται από ένα λεπτό πλαστικό υπόστρωμα που επιχρυσώνεται και μετά καλύπτεται με στρώμα βαφής. Η βαφή είναι χρυσή, πράσινη ή μπλέ και έχει την ιδιότητα να αλλάζει κατάσταση όταν υπόκειται στην ακτινοβολία μιας ισχυρής δέσμης laser. Η αλλαγή κατάστασης της βαφής είναι μόνιμη, με αποτέλεσμα τα CD-R να μπορούν να εγγραφούν μια φορά μόνο. Στη μια κατάσταση το στρώμα βαφής είναι σχεδόν διαφανές ή υπέρυθρο[1], ενώ.στην άλλη είναι σχεδόν αδιαφανές. Η οπτική ακτίνα χαμηλής ισχύος που χρησιμοποιείται κατά την ανάγνωση, διαπερνά το στρώμα βαφής, ανακλάται από το στρώμα χρυσού και επιστρέφει πίσω στο φωτοευαίσθητο κύτταρο. Ανάλογα με την κατάσταση του στρώματος βαφής, η φωτεινή δέσμη ανακλάται με διαφορετική ισχύ, επομένως διαβάζεται το 0 ή το 1.

Επανεγγράψιμα CD (CD-RW)

Σε αντίθεση με τα εγγράψιμα CD, όπου το στρώμα εγγραφής καλύπτεται από βαφή που μπορεί να αλλάξει κατάσταση μόνο μια φορά, τα επανεγγράψιμα CD αποτελούνται από υλικό που μπορεί να αλλάξει κατάσταση πολλές φορές. Το στρώμα εγγραφής αποτελείται από ένα χρυσόχρωμο κράμα που μετατρέπεται από άμορφο σε κρυσταλλικό όταν εκτεθεί σε ισχυρή πηγή φωτός laser. Η αλλαγή κατάστασης του υλικού παραμένει ακόμα και όταν πάψει να υπάρχει η επίδραση της ακτινοβολίας, ενώ επιστρέφει στην προηγούμενη κατάσταση όταν εκτεθεί σε ακτινοβολία διαφορετικής συχνότητας. Η κρυσταλλική μορφή του κράματος ανακλά το φως όπως οι λείες επιφάνειες στα απλά CD. Η άμορφη κατάστασή του λειτουργεί όπως οι κοιλότητες στα απλά CD. Με δύο ακτινοβολίες laser διαφορετικής συχνότητας λοιπόν μπορούμε να προκαλούμε αλλαγές κατάστασης στην επιφάνεια του δίσκου, δηλαδή διαδοχικές επανεγγραφές. Οι αλλαγές κατάστασης από άμορφη σε κρυσταλλική και αντίστροφα δεν μπορούν να επαναλαμβάνονται επ' άπειρο. Ο αριθμός των εγγραφών σε ένα CD-RW είναι περιορισμένος, συνήθως γύρω στις 1000 εγγραφές.

DVD

Ο *δίσκοι DVD* (Digital Versatile Disk) αποτελούν την επόμενη γενιά των οπτικών δίσκων. Στην πράξη είναι μεγαλύτεροι, γρηγορότεροι δίσκοι CD που περιέχουν μέχρι 4,7 GB για κάθε στρώμα και πλευρά εγγραφής. Η χωρητικότητά τους φθάνει μέχρι τα 17 GB για δίσκους που έχουν εγγραφεί και από τις δύο πλευρές.

Κάθε πλευρά ενός δίσκου DVD αποτελείται από δύο στρώματα. Η δέσμη ανάγνωσης διαβάζει και τα δύο αυτά στρώματα· έτσι περιέχεται διπλάσια πληροφορία σε κάθε πλευρά.

Οι κυριότερες κατηγορίες των δίσκων DVD είναι:

⇨ **DVD-Video**: Οι δίσκοι DVD-Video μπορούν να αποθηκεύσουν ολόκληρες κινηματογραφικές ταινίες για να αναπαραχθούν στην οθόνη ενός προσωπικού υπολογιστή. Οι ταινίες είναι κωδικοποιημένες και αποθηκευμένες σε μορφή

[1] Δηλαδή αντανακλά ακτινοβολίες που η συχνότητά τους τις τοποθετεί στο υπέρυθρο τμήμα του φάσματος.

MPEG-2. Για την αναπαραγωγή τους, εκτός από τη συσκευή ανάγνωσης δίσκων DVD, απαιτείται και το υποσύστημα αποκωδικοποίησης MPEG-2 που συνδέεται με τη μορφή κάρτας στο διάδρομο του υπολογιστή. Το DVD-Video έρχεται να αντικαταστήσει το κλασικό οικιακό video, υποστηρίζοντας επιπλέον προχωρημένες τεχνικές αναπαραγωγής σε σχέση με τις κλασικές ταινίες VHS. Στο δίσκο αποθηκεύονται ξεχωριστά η ταινία χωρίς ήχο και η ηχητική επένδυση, ώστε να μπορεί κανείς να διαλέξει τους υπότιτλους ή τους διαλόγους στη γλώσσα που επιθυμεί. Επίσης μπορεί κανείς να παρακολουθήσει την ίδια σκηνή από διαφορετικές οπτικές γωνίες. Οι δίσκοι DVD αποτελούν την εξέλιξη του CD-I (interactive CD), προτύπου που ανέπτυξε η Philips το 1987 για την αναπαραγωγή ταινιών σε κωδικοποίηση Mpeg και την απευθείας σύνδεση με την οικιακή τηλεόραση.

- **DVD-ROM**: Οι δίσκοι DVD, πέρα από τη χρήση τους ως μέσα αποθήκευσης και αναπαραγωγής ταινιών και μουσικής, μπορούν να αποθηκεύσουν και μεγάλο όγκο ψηφιακών δεδομένων. Στο άμεσο μέλλον αναμένεται να αντικαταστήσουν τους οπτικούς δίσκους CD των 650MB.
- **DVD-RAM και DVD-RW+**: Η τεχνολογία εγγραφής δίσκων DVD ονομάζεται DVD-RAM και DVD-RW+. Στην πραγματικότητα πρόκειται για δυο πρότυπα από διαφορετικές εταιρείες για επανεγγράψιμους δίσκους DVD που λειτουργούν ανάλογα με τα επανεγγράψιμα CD-RW.

Ανακεφαλαίωση

Οι συσκευές ανάγνωσης οπτικών δίσκων βασίζονται στην ανάκλαση μιας φωτεινής δέσμης laser από την διαμόρφωση της επιφάνειας του οπτικού δίσκου. Οι πιο διαδεδομένες κατηγορίες οπτικών δίσκων είναι τα CD-ROM (μουσικά CD και CD δεδομένων), εγγράψιμα CD, επανεγγράψιμα CD και DVD. Η τεχνολογία εγγραφής βασίζεται στη δημιουργία μόνιμης ή προσωρινής αλλαγής στο υλικό του οπτικού δίσκου (στρώμα βαφής ή κρυσταλλικό κράμα).

Οι δίσκοι DVD έχουν πολύ μεγαλύτερη χωρητικότητα από τους κοινούς οπτικούς δίσκος και χρησιμοποιούνται κυρίως σε εφαρμογές πολυμέσων, όπως αποθήκευση ταινιών, μουσικής, αλλά και σε εφαρμογές δημιουργίας αντιγράφων εφεδρείας (κυρίως οι επανεγγράψιμοι DVD).

Γλωσσάριο όρων

CD Δεδομένων	*CD-Data*
DVD	*Digital Versatile Disk*
Εγγράψιμα CD	*CD-Recordable ή WORM-Write Once Read Many*
Επανεγγράψιμα CD	*CD-RW Rewritable ή CD-Erasable*
Κοιλότητα ή Εσοχή	*Pit*
Μουσικά CD	*CD-Audio*
Οπτικός Δίσκος	*Compact Disks*

Ερωτήσεις

? Περίγραψε τον τρόπο εγγραφής και ανάγνωσης στα CD-audio.

? Περίγραψε τον τρόπο εγγραφής και ανάγνωσης στα CD-ROM.

? Περίγραψε τον τρόπο εγγραφής και ανάγνωσης στα επανεγράψιμα CD.

Μάθημα 5.6 Συσκευές Τηλεπικοινωνιών και Δικτύωσης

Σκοπός του μαθήματος

Σκοπός του μαθήματος είναι να εισαγάγει το μαθητή στις βασικές τεχνικές και στις συσκευές που χρησιμοποιούνται για την επικοινωνία των υπολογιστών και τη διασύνδεση τους σε δίκτυα.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς πώς λειτουργεί ένα modem*
- *Να περιγράφεις την κάρτα δικτύου*
- *Να απαριθμείς τις κυριότερες συσκευές υποστήριξης δικτύων*
- *Να απαριθμείς τα φυσικά μέσα διασύνδεσης που χρησιμοποιούνται για την επικοινωνία των υπολογιστών*

Η τεχνολογία που έχει αυξήσει ραγδαία τη χρήση των ηλεκτρονικών υπολογιστών είναι η δικτύωσή τους, δηλαδή η διασύνδεσή τους μέσω κάποιου φυσικού μέσου για την ανταλλαγή δεδομένων και την από κοινού χρήση των περιφερειακών τους μονάδων. Στις μέρες μας οι υπολογιστές διασυνδέονται με διάφορους τρόπους, ανάλογα με το χρησιμοποιούμενο μέσο και τη φυσική τους απόσταση. Οι περισσότεροι υπολογιστές αποτελούν τμήματα ενός μικρού ή μεγαλύτερου δικτύου, τοπικού ή ευρείας περιοχής. Διασυνδέονται με μέσα όπως οι τηλεφωνικές γραμμές, οι οπτικές ίνες, με ασύρματο τρόπο ή ακόμα και μέσω δορυφόρων.

Συσκευές Δικτύωσης

Παράλληλη-Σειριακή Θύρα

Ο πιο απλός τρόπος διασύνδεσης δύο υπολογιστών είναι μέσω ενός καλωδίου που συνδέει τις αντίστοιχες παράλληλες ή σειριακές θύρες τους. Η *παράλληλη θύρα* ή θύρα centronics, όπως λέγεται, έχει 25 ακίδες, και μεταδίδει παράλληλα 8 bits δεδομένων. Η σειριακή θύρα ή RS-232 έχει 9 ή 25 ακίδες, μεταδίδει ένα bit κάθε φορά, και φθάνει σε ταχύτητα μετάδοσης μέχρι τα 115.200 bits το δευτερόλεπτο.

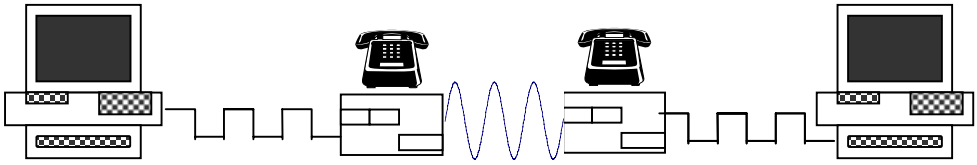
Η σύνδεση μέσω των θυρών αυτών περιορίζεται από την απόσταση, καθώς τα καλώδια δεν μπορούν να είναι μεγαλύτερα των 10 m (για την παράλληλη θύρα). Αποτελεί όμως την πιο απλή λύση για ανταλλαγή δεδομένων μεταξύ δύο υπολογιστών.

Modem

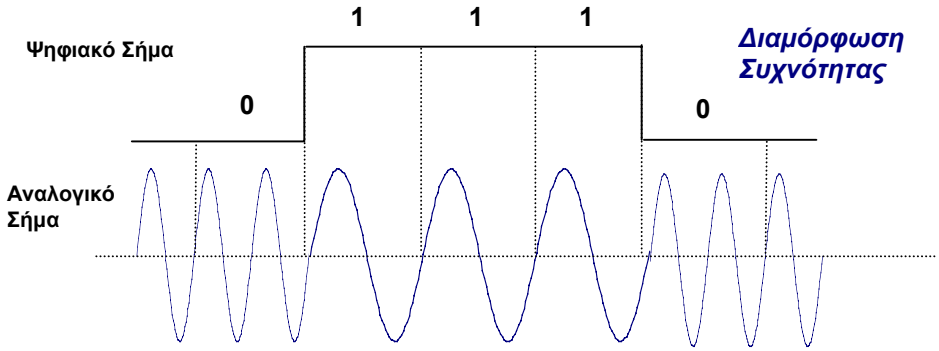
Ο πιο συνηθισμένος τρόπος διασύνδεσης δύο υπολογιστών είναι μέσω μίας τηλεφωνικής γραμμής. Από νωρίς χρησιμοποιήθηκε η τεχνολογία των κοινών τηλεφωνικών δικτύων, καθώς το τηλέφωνο είναι η πιο διαδεδομένη συσκευή

επικοινωνίας. Βέβαια οι τηλεφωνικές γραμμές μεταδίδουν ηλεκτρικό σήμα που αντιστοιχεί στη φωνή μας, οπότε χρειάζεται η μετατροπή των ψηφιακών δεδομένων των υπολογιστών σε σήμα αναλογικής μορφής που μεταδίδεται μέσω των τηλεφωνικών γραμμών.

Η συσκευή που μετατρέπει τη σειρά των ψηφιακών δεδομένων σε αναλογικό σήμα που να μπορεί να διαδίδεται μέσω των τηλεφωνικών γραμμών και αντίστροφα, ονομάζεται modem. Το modem μεταδίδει ένα σταθερό ημιτονοειδές αναλογικό σήμα, το *φέρον* σήμα (carrier). Τα χαρακτηριστικά του φέροντος, δηλαδή το *πλάτος*, η *συχνότητα* και η *φάση* του μεταβάλλονται ανάλογα με την τιμή της ψηφιακής πληροφορίας (το 0 ή 1). Με άλλα λόγια γίνεται μια κωδικοποίηση του ψηφιακού σήματος σε αναλογικό σήμα, με την πληροφορία να «μεταφέρεται» μέσω του φέροντος σήματος. Το modem αναλαμβάνει στη μία άκρη τη διαμόρφωση του αναλογικού σήματος και στην άλλη την αποδιαμόρφωση του αναλογικού ήχου που λαμβάνεται, ώστε να φτάσει στον παραλήπτη-υπολογιστή η ψηφιακή πληροφορία.



Η τεχνική αυτή *διαμόρφωσης-αποδιαμόρφωσης* (Modulation-Demodulation) είναι μια ευρέως διαδεδομένη τεχνική στην επιστήμη των τηλεπικοινωνιών, όχι μόνο στους υπολογιστές. Σε οποιοδήποτε φυσικό μέσο μετάδοσης ενός σήματος, π.χ. τον αέρα, την οπτική ίνα, το χάλκινο καλώδιο, φροντίζουμε να «κωδικοποιούμε» τη χρήσιμη πληροφορία που θέλουμε να μεταδώσουμε πάνω σε ένα σήμα που μπορεί να μεταφερθεί καλύτερα (π.χ. με τις λιγότερες απώλειες σε ισχύ) στο φυσικό μέσο. Αυτή η αρχή εφαρμόζεται στα ραδιοφωνικά σήματα (διαμόρφωση AM ή FM), όπου ένα αναλογικό σήμα διαμορφώνει ένα άλλο αναλογικό σήμα μεγαλύτερης συχνότητας.



Οι βασικοί τρόποι διαμόρφωσης είναι:

Διαμόρφωση κατά πλάτος (AM-Amplitude Modulation): Το πλάτος του φέροντος σήματος μεταβάλλεται ανάλογα με το σήμα της πληροφορίας.

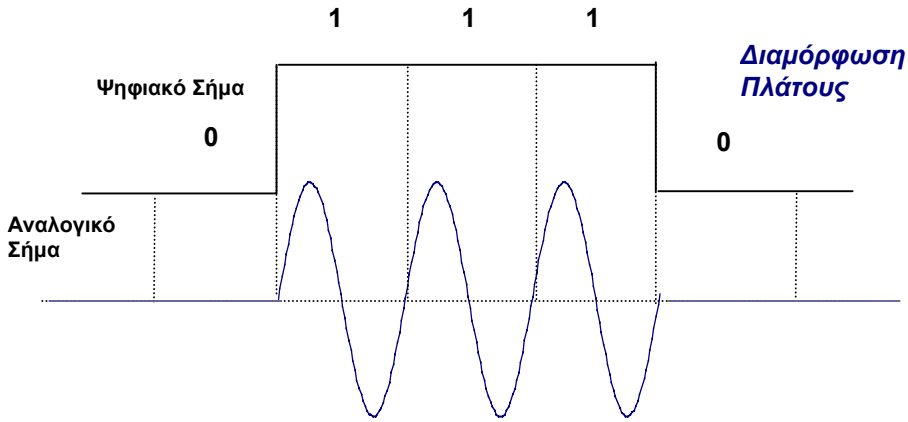
Διαμόρφωση κατά συχνότητα (FM-Frequency Modulation): Στην περίπτωση αυτή η συχνότητα του φέροντος σήματος μεταβάλλεται ανάλογα με το σήμα της πληροφορίας.

Διαμόρφωση κατά φάση (Phase Modulation): Η φάση του φέροντος σήματος μεταβάλλεται ανάλογα με την πληροφορία.

Η ταχύτητα μετάδοσης στη διασύνδεση δύο υπολογιστών μετριέται σε bits ανά δευτερόλεπτο. Εκφράζει το ρυθμό που μεταδίδονται τα ψηφιακά δεδομένα από τον έναν υπολογιστή στον άλλο. Γενικότερα οι ταχύτητες των δικτύων εκφράζονται σε Kbits/sec ή και Mbits/sec. Στην περίπτωση της διασύνδεσης μέσω τηλεφωνικών γραμμών υπάρχει και άλλη μια μονάδα μέτρησης της ταχύτητας, πέρα από το ρυθμό μετάδοσης, που ίσως δημιουργεί σύγχυση. Είναι η ταχύτητα μεταβολής των χαρακτηριστικών του φέροντος σήματος στην τηλεφωνική γραμμή, και μετριέται σε

baud. Για παράδειγμα, μετράμε σε baud, το ρυθμό εναλλαγών από τη μια συχνότητα στην άλλη για την περίπτωση FM.

Ας υποθέσουμε ότι χρησιμοποιούμε δύο διαφορετικές συχνότητες για το φέρον σήμα και αντιστοιχούμε στη φάση της διαμόρφωσης το μηδέν στη μια και το ένα στην άλλη. Τότε η ταχύτητα μετάδοσης σε bits θα ήταν ίση με την ταχύτητα σε baud. Αν όμως χρησιμοποιούμε για το φέρον 4 διαφορετικές συχνότητες, σε κάθε συχνότητα μπορούμε να αντιστοιχίσουμε 2 bits. Έτσι, αν το φέρον έχει ταχύτητα 1200 baud, και για κάθε baud έχουμε 2 bit, η ταχύτητα μετάδοσης των δεδομένων είναι 2400 bits/sec.



Η ταχύτητα μεταβολής του φέροντος περιορίζεται από την ποιότητα του φυσικού μέσου, δηλαδή την τεχνολογία των τηλεφωνικών γραμμών. Επίσης δεν μπορούμε να χρησιμοποιήσουμε πάρα πολλές στάθμες διαμόρφωσης (πλάτη ή συχνότητες), γιατί πρέπει να γίνεται χωρίς λάθη η αποδιαμόρφωση. Αυτό σημαίνει ότι οι διαφορετικές στάθμες του σήματος πρέπει να είναι διακεκριμένες. Η τεχνολογία των modem λοιπόν επιτρέπει σχετικά χαμηλές ταχύτητες διασύνδεσης. Τα πρώτα modem συνέδεαν υπολογιστές με ταχύτητες 300 έως και 2400 bit/sec ενώ στις μέρες μας φθάνουν μέχρι τα 56.600 bits/sec. Σε ιδιαίτερες συνθήκες μετάδοσης (καλή ποιότητα γραμμών, κοντινές αποστάσεις) οι ταχύτητες αυξάνονται (π.χ. baseband modems)

Κάρτες δικτύου

Οι κάρτες δικτύου αποτελούν τις συσκευές μέσω των οποίων γίνεται η διασύνδεση ενός υπολογιστή σε ένα τοπικό δίκτυο. Η κάρτα δικτύου συνδέεται στο διάδρομο του υπολογιστή και αναλαμβάνει να μεταφέρει δεδομένα από και προς το δίκτυο. Ο ρόλος της είναι να ομαδοποιεί τα δεδομένα, μετατρέποντάς τα σε κάποια μορφή που είναι συμβατή με το πρωτόκολλο του δικτύου, ώστε να μπορούν να διαβαστούν από τους υπόλοιπους υπολογιστές. Τα ηλεκτρονικά κυκλώματα των καρτών αναλαμβάνουν τη διόρθωση των λαθών, την αποστολή και λήψη των δεδομένων. Το πιο διαδεδομένο πρωτόκολλο δικτύωσης είναι το Ethernet, και οι περισσότερες κάρτες στους προσωπικούς υπολογιστές λειτουργούν με αυτήν την κατηγορία δικτύων.

Οι κάρτες δικτύου Ethernet στο πίσω μέρος τους έχουν ειδικές υποδοχές για διαφόρων ειδών συνδέσεις: υπάρχει η υποδοχή για συνδεσμολογία *ομοαξονικού καλωδίου* (BNC) και η υποδοχή για συνδεσμολογία *συνεστραμμένου ζεύγους* (RJ45).

Επαναλήπτες - Δρομολογητές - Γέφυρες

Οι συσκευές αυτές χρησιμοποιούνται για να υποστηρίζουν τα δίκτυα των υπολογιστών. Ένας *επαναλήπτης* (repeater) είναι μια συσκευή ενίσχυσης του σήματος που ταξιδεύει μέσα στο φυσικό μέσο δικτύωσης, για να μπορούμε να δημιουργούμε δίκτυα μεταξύ υπολογιστών που είναι απομακρυσμένοι μεταξύ

τους περισσότερο απ' όσο ορίζουν οι προδιαγραφές ενός συγκεκριμένου μέσου μετάδοσης. Όσο μακρύτερο είναι π.χ. ένα καλώδιο, τόσο πέφτει η ισχύς του σήματος και χρειάζεται ενίσχυση για να μεταδοθεί σε μεγάλες αποστάσεις. Ο επαναλήπτης τοποθετείται σε τακτά διαστήματα κατά μήκος της διεύθυνσης μετάδοσης και ενισχύει το σήμα.

Ο *δρομολογητής* (router) και η *γέφυρα* (bridge) φροντίζουν για τη συνένωση διαφορετικών δικτύων. Ο ρόλος τους είναι να προωθούν τα πακέτα πληροφορίας από το ένα δίκτυο στο άλλο επιλεκτικά, ανάλογα με τον υπολογιστή-προορισμό. Έτσι αποφεύγεται η άσκοπη μετάδοση πακέτων δεδομένων σε τμήματα δικτύων που είναι άσχετα με τον προορισμό.

Πρωτόκολλα Επικοινωνίας

Η δικτύωση μας δίνει τη δυνατότητα διασύνδεσης υπολογιστών με διαφορετικά λειτουργικά συστήματα και επικοινωνίας μεταξύ διαφορετικών εφαρμογών λογισμικού. Για να γίνει εφικτή η επικοινωνία, όλοι οι υπολογιστές και οι εφαρμογές χρησιμοποιούν το ίδιο σύνολο κανόνων και διαδικασιών για την ανταλλαγή δεδομένων. Αυτοί οι κανόνες ονομάζονται *πρωτόκολλο επικοινωνίας* (communication protocol). Έτσι τα δεδομένα μπορούν να αναπαρίστανται διαφορετικά σε κάθε υπολογιστή, όμως στο δίκτυο «ταξιδεύουν» έχοντας μορφή και οργάνωση κοινά αποδεκτές απ' όλους τους υπολογιστές του δικτύου.

Εξαιτίας της πληθώρας διαφορετικών αρχιτεκτονικών υπολογιστών, λειτουργικών συστημάτων, εφαρμογών και τύπων δικτύων υπάρχει πολυπλοκότητα στο είδος και στο πλήθος των πρωτοκόλλων επικοινωνίας. Υπάρχουν π.χ. πρωτόκολλα που ρυθμίζουν την επικοινωνία στο φυσικό μέσο διασύνδεσης, μεταξύ των καρτών δικτύου και του καλωδίου. Σε αυτά τα πρωτόκολλα δε γίνεται διάκριση μεταξύ των πληροφοριών · όλες μαζί ενσωματώνονται σε πακέτα, δηλαδή αυτοτελείς μονάδες πληροφορίας μήκους μερικών εκατοντάδων bytes που ταξιδεύουν στο δίκτυο.

Τα πρωτόκολλα εξαρτώνται από την ταχύτητα του δικτύου, τον τρόπο επικοινωνίας υπολογιστή - δικτύου κλπ. Υπάρχουν όμως πρωτόκολλα ανωτέρου επιπέδου που έχουν σχέση με την επικοινωνία των εφαρμογών και διακρίνουν τις πληροφορίες, μεταφράζοντάς τις για επεξεργασία από κάποιο πρόγραμμα. Το κάθε πρωτόκολλο έχει τη δυνατότητα να ελέγχει και να διορθώνει λάθη στο επίπεδο που ρυθμίζει.

Τα πιο γνωστά πρωτόκολλα επικοινωνίας είναι:

- ☑ **Xmodem/Zmodem/Ymodem/Kermit:** Απλά πρωτόκολλα επικοινωνίας για σύνδεση μέσω modem. Διαφέρουν ως προς το μέγεθος του πακέτου που χρησιμοποιείται για την αποστολή δεδομένων. Έτσι το Xmodem χρησιμοποιεί μέγεθος 128 bytes για κάθε πακέτο, to Zmodem 512 bytes και το Ymodem 1024 bytes. Το Kermit έχει πακέτα μεταβλητού μεγέθους.
- ☑ **Ethernet**: Το πιο διαδεδομένο πρωτόκολλο επικοινωνίας σε τοπικά δίκτυα υπολογιστών
- ☑ **Token Ring:** Πρωτόκολλο επικοινωνίας που βασίζεται στην ύπαρξη ενός ειδικού πακέτου (σκυτάλη) που ταξιδεύει μέσα στο δίκτυο. Όποιος υπολογιστής είναι ιδιοκτήτης του πακέτου, έχει τον έλεγχο του δικτύου, και μπορεί να αποστείλει δεδομένα.
- ☑ **FDDI:** Πρωτόκολλο επικοινωνίας για μεγάλες ταχύτητες δικτύων με οπτικές ίνες. Χρησιμοποιείται κυρίως σε *δίκτυα κορμού* (backbone) που συνενώνουν μικρότερα δίκτυα Ethernet.
- ☑ **ATM:** Πρωτόκολλο που υποστηρίζει πολύ μεγάλες ταχύτητες επικοινωνίας
- ☑ **Frame Relay**: Άλλο ένα πρωτόκολλο υψηλών ταχυτήτων για δίκτυα κορμού.

- ☑ **PowerTalk:** Πρωτόκολλο επικοινωνίας για υπολογιστές Apple Macintosh
- ☑ **X.25:** Διεθνές πρότυπο για δίκτυα μεταγωγής πακέτου.
- ☑ **TCP/IP:** Πρωτόκολλο περιγραφής των πακέτων που μεταδίδονται σε ένα δίκτυο. Είναι το ευρέως διαδεδομένο πρωτόκολλο στην κοινότητα του Διαδικτύου (Internet).
- ☑ **IPX:** Πρωτόκολλο με λειτουργία αντίστοιχη του TCP/IP που χρησιμοποιείται σε δίκτυα Novell.

Φυσικά μέσα διασύνδεσης

Τα μέσα διασύνδεσης είναι τα φυσικά υλικά ή άλλα μέσα που χρησιμοποιούνται στην δημιουργία των δικτύων επικοινωνίας. Οι δύο κύριοι τύποι των μέσων διασύνδεσης είναι τα **ενσύρματα μέσα** όπως είναι τα συνεστραμμένα καλώδια, τα ομοαξονικά και οι οπτικές ίνες και τα **ασύρματα μέσα** όπως είναι τα μικροκύματα, τα ραδιοκύματα και η υπεριώδης ακτινοβολία.

➲ Το *ζεύγος συνεστραμμένων καλωδίων* (twisted-pair) αποτελείται από σύρματα χαλκού επικαλυμμένα με πλαστικό τα οποία συστρέφονται μαζί. Ένα λεπτό στρώμα από έγχρωμο πλαστικό απομονώνει και ξεχωρίζει τα σύρματα που στη συνέχεια συστρέφονται προκειμένου να μειωθεί η ηλεκτρική αλληλεπίδραση τους. Τα καλώδια twisted-pair χωρίζονται:

(i) Στα *προστατευμένα καλώδια twisted-pair* (STP)· σε αυτά γύρω από το κάθε σύρμα τυλίγεται ένα φύλλο μετάλλου, το οποίο μειώνει επιπλέον την εξωτερική ηλεκτρική επίδραση.

(ii) Στα *απροστάτευτα καλώδια twisted-pair* (UTP), που ονομάζονται επίσης και 10BaseT καλώδια. Σε αυτά δεν υπάρχει το επιπλέον φύλλο μετάλλου.

Επειδή τα συνεστραμμένα καλώδια είναι ένα μέσο μετάδοσης μικρού κόσοτυς χρησιμοποιούνται συνήθως τόσο στις τηλεφωνικές γραμμές όσο και για τη μετάδοση δεδομένων μεταξύ υπολογιστή και τερματικών.

➲ Το *ομοαξονικό* (coaxial) καλώδιο αποτελείται από ένα χάλκινο σύρμα περιβαλλόμενο από τρία στρώματα: μια μόνωση, έναν εξωτερικό μεταλλικό αγωγό και ένα πλαστικό περίβλημα. Επειδή το ομοαξονικό καλώδιο είναι περισσότερο μονωμένο από ότι το συνεστρεμμένο καλώδιο, συνιστάται για μεγάλες αποστάσεις όσο αφορά την ηλεκτρική αλληλεπίδραση και μεταφορά δεδομένων με ταχύτητα.

➲ Η *οπτική ίνα* (fiber optic) είναι μια εύκαμπτη, λεπτή σαν τρίχα, ίνα από γυαλί ή πλαστικό για τη μεταφορά δεδομένων με τη μορφή παλμών φωτός. Το σημαντικό πλεονέκτημα της οπτικής ίνας σε σχέση με τα άλλα καλώδια είναι το μικρότερο βάρος της, ο χώρος που εξοικονομεί και η απουσία ηλεκτρικής και μαγνητικής επίδρασης σε αυτήν. Συνεπώς μπορούν να μεταδώσουν πληροφορία σε μεγάλες αποστάσεις χωρίς την ανάγκη ενίσχυσης του σήματος. Ένα ακόμη πλεονέκτημά της είναι η υψηλή ταχύτητα μετάδοσης. Μία μόνο οπτική ίνα μπορεί να μεταφέρει αρκετές εκατοντάδες χιλιάδες παλμούς στη μονάδα του χρόνου. Οι οπτικές ίνες ωστόσο κοστίζουν πολύ περισσότερο από τα διάφορα καλώδια και επιπλέον υπάρχουν δυσκολίες στην εγκατάσταση και τη λειτουργία τους. Παρά τους προηγούμενους περιορισμούς, πολλές τηλεφωνικές εταιρείες χρησιμοποιούν οπτικές ίνες για νέες τηλεφωνικές γραμμές.

➲ Τα *μικροκύματα* (microwaves), είναι ραδιοκύματα που χρησιμοποιούνται για την μετάδοση τόσο αναλογικών όσο και ψηφιακών σημάτων με υψηλή ταχύτητα. Τα *επίγεια* μικροκύματα (terrestrial microwaves), χρησιμοποιούνται για τη μετάδοση από ένα σταθμό μικροκυμάτων σε άλλο, περίπου όπως μεταφέρονται τα σήματα

των ραδιοφωνικών σταθμών. Τα μικροκύματα περιορίζονται από τη διαδρομή μετάδοσης, γιατί πρέπει να μεταδίδονται απευθείας χωρίς να παρεμποδίζονται από κτίρια ή βουνά ανάμεσα στους σταθμούς μετάδοσης. Για να αποφευχθεί αυτή η παρεμπόδιση, οι κεραίες μικροκυμάτων συνήθως τοποθετούνται στην κορυφή κτιρίων, πύργων ή βουνών.

➲ Οι *τηλεπικοινωνιακοί δορυφόροι* (telecommunication satellites) λαμβάνουν τα μικροκύματα από τις επικοινωνιακές εγκαταστάσεις εδάφους, τα αποκωδικοποιούν και τα ξαναστέλνουν σε άλλες επικοινωνιακές εγκαταστάσεις. Οι εγκαταστάσεις εδάφους ονομάζονται *επίγειοι σταθμοί* (terrestrial stations) και χρησιμοποιούν μεγάλες κεραίες σε σχήμα δίσκου για να λαμβάνουν και να στέλνουν δεδομένα στους δορυφόρους. Η μετάδοση προς το δορυφόρο ονομάζεται *uplink* και η λήψη απ' αυτόν *downlink.*

Ανακεφαλαίωση

Οι υπολογιστές μπορούν να συνδεθούν μεταξύ τους μέσω της παράλληλης ή της σειριακής θύρας τους, μέσω τηλεφωνικής γραμμής με τη βοήθεια του modem, ή σε ένα δίκτυο μέσω μίας κάρτας δικτύου. Οι κανόνες επικοινωνίας μεταξύ των υπολογιστών σε ένα δίκτυο ονομάζονται πρωτόκολλο. Το πιο γνωστό πρωτόκολλο για τοπικά δίκτυα είναι το Ethernet. Η φυσική σύνδεση των υπολογιστών μπορεί να γίνει με διάφορα μέσα, ενσύρματα (καλώδια και οπτικές ίνες) ή ασύρματα (μικροκύματα ή υπεριώδη ακτινοβολία).

Γλωσσάριο όρων

Downlink	
Modem	
Uplink	
Απροστάτευτα Καλώδια Twisted-Pair	*UTP*
Γέφυρα	*Bridge*
Διαμόρφωση Κατά Πλάτος	*Amplitude Modulation*
Διαμόρφωση Κατά Συχνότητα	*Frequency Modulation*
Διαμόρφωση Κατά Φάση	*Phase Modulation*
Διαμόρφωση-Αποδιαμόρφωση	*Modulation-Demodulation*
Δίκτυο Κορμού	*Backbone*
Διπλά Συνεστραμμένο Καλώδιο	*Twisted-Pair*
Δρομολογητής	*Router*
Επαναλήπτης	*Repeater*
Επίγεια Μικροκύματα	*Terrestrial Microwaves*
Μικροκύματα	*Microwaves*
Ομοαξονικό Καλώδιο	*BNC*
Οπτική Ίνα	*Fiber Optic*
Παράλληλη Θύρα	*Parallel Port - Centronics*
Προστατευμένα Καλώδια Twisted-Pair	*STP*
Πρωτόκολλο Επικοινωνίας	*Communication Protocol*
Τηλεπικοινωνιακός Δορυφόρος	*Telecommunication Satellite*

Ερωτήσεις

? Περίγραψε τη λειτουργία ενός modem.

? Πώς λειτουργεί μια κάρτα δικτύου;

? Τι πλεονεκτήματα έχουν οι οπτικές ίνες σαν μέσο μετάδοσης;

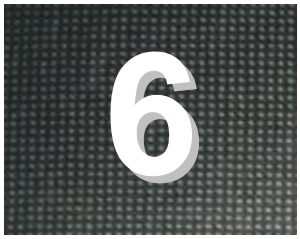
Τι μάθαμε σε αυτό το κεφάλαιο

- Ο διάδρομος αποτελεί το μέσο διασύνδεσης των επιμέρους συστημάτων που αποτελούν τον υπολογιστή.
- Τα βασικότερα χαρακτηριστικά των εκτυπωτών είναι η ταχύτητά τους και η ανάλυση εκτύπωσης. Ως προς την τεχνολογία τους, οι εκτυπωτές διακρίνονται σε κρουστικούς, έκχυσης (ψεκασμού) μελάνης και εκτυπωτές laser. Ως προς τον τρόπο επεξεργασίας της εκτύπωσης διακρίνονται σε εκτυπωτές γραμμής και εκτυπωτές σελίδας.
- Η πιο διαδεδομένη κατηγορία οθονών υπολογιστών είναι οι οθόνες καθοδικού σωλήνα. Η λειτουργία τους στηρίζεται στην ύπαρξη ενός πυροβόλου που εκτοξεύει ηλεκτρόνια σε φωσφορίζουσα επιφάνεια προκαλώντας την ακτινοβολία της. Οι οθόνες υγρών κρυστάλλων στηρίζονται στο φαινόμενο της πόλωσης του φωτός όταν διέρχεται από κρυσταλλική δομή.
- Οι μαγνητικοί δίσκοι αποτελούν το κύριο μέσο αποθήκευσης στους υπολογιστές. Η εγγραφή τους βασίζεται στη μαγνήτιση της επιφάνειας του δίσκου, και η ανάγνωση στη δημιουργία επαγωγικού ρεύματος ορθής ή ανάστροφης φοράς κατά το πέρασμα ενός ηλεκτρομαγνήτη πάνω από τη μαγνητισμένη επιφάνεια
- Η τεχνολογία ανάγνωσης οπτικών δίσκων βασίζεται στην ανάκλαση μιας φωτεινής δέσμης laser εξαιτίας της διαμόρφωση της επιφάνειας του οπτικού δίσκου.
- Οι υπολογιστές μπορούν να συνδεθούν μεταξύ τους μέσω της παράλληλης ή της σειριακής θύρας τους, μέσω τηλεφωνικής γραμμής με τη βοήθεια του modem, ή σε ένα δίκτυο μέσω μίας κάρτας δικτύου. Οι κανόνες επικοινωνίας μεταξύ των υπολογιστών σε ένα δίκτυο ονομάζονται πρωτόκολλο.

Βιβλιογραφία – Πηγές

- Luce T., Αρχιτεκτονική Υπολογιστών, Τζιόλας, 1991.
- Maran R. Μαθαίνω μόνος μου τους υπολογιστές και το Internet, Γεννάδειος Σχολή, 1995.
- Πώς δουλεύουν οι Υπολογιστές, Κλειδάριθμος, 1994.
- Hutchinson S., Sawyer S., Computers and Information Systems, Irwin, 1996.
- Vermaat M., Walker T., Hall T., Repede J., Discovering Computer, Shelly Cashman Series, 1997.

Κεφάλαιο



Ο Προσωπικός Υπολογιστής

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει τον πιο διαδεδομένο υπολογιστή, τον Προσωπικό Υπολογιστή και να περιγράψει την αρχιτεκτονική και τη μορφή του.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις τη γενική αρχιτεκτονική του Προσωπικού Υπολογιστή.
- Να αναγνωρίζεις τα κυριότερα τμήματά του και τα σημεία όπου συνδέονται αυτά στη μητρική κάρτα.
- Να απαριθμείς τα διάφορα είδη και τύπους μνήμης σε ένα Προσωπικό Υπολογιστή.

Μαθήματα

6.1 Ιστορία και Αρχιτεκτονική του Προσωπικού Υπολογιστή

6.2 Η Μητρική Κάρτα του Προσωπικού Υπολογιστή

6.3 Η Μνήμη στον Προσωπικό Υπολογιστή

Μάθημα 6.1 Ιστορία και Αρχιτεκτονική του Προσωπικού Υπολογιστή

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει την ιστορία του προσωπικού υπολογιστή και να περιγράψει τη γενική αρχιτεκτονική του.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να αφηγείσαι συνοπτικά την ιστορία του προσωπικού υπολογιστή*
- *Να περιγράφεις τη λειτουργία του διαδρόμου του*
- *Να εξηγείς τη γενική λειτουργία των επεξεργαστών που χρησιμοποιούν οι προσωπικοί υπολογιστές*
- *Να απαριθμείς διάφορους τύπους περιφερειακής μνήμης για προσωπικούς υπολογιστές*

Ο πρώτος *προσωπικός υπολογιστής* (personal computer, PC) παρουσιάστηκε από την εταιρεία IBM το 1981. Περιείχε τον επεξεργαστή 8088 της εταιρείας Intel σε ταχύτητα 4,77 MHz, 16 Kb κύριας μνήμης που μπορούσε να επεκταθεί ως τα 256 Kb, ασπρόμαυρη οθόνη και μία ή δύο μονάδες δισκέτας 5,25" των 160 Kb. Ο νέος αυτός υπολογιστής μπήκε δυναμικά στην αγορά των μικροϋπολογιστών με πωλήσεις 136.000 IBM PC μέσα στον πρώτο ενάμιση χρόνο της κυκλοφορίας του.

Η μεγάλη επιτυχία του IBM PC έκανε τους δημιουργούς του να κατασκευάσουν παραλλαγές του, που ονομάστηκαν XT και AT. Συγχρόνως, άλλες εταιρείες άρχισαν να κατασκευάζουν απομιμήσεις του που αναφέρονταν ως «συμβατοί με IBM». Με την πάροδο του χρόνου, η βασική αρχιτεκτονική του υπολογιστή αυτού παγιώθηκε, όπως και οι επεξεργαστές της εταιρείας Intel ως η βασική δομική του μονάδα.

Αφού η αρχιτεκτονική του προσωπικού υπολογιστή ήταν πλέον δεδομένη, άνοιξε ο δρόμος για το σχεδιασμό μιας ποικιλίας περιφερειακών συσκευών και άλλων τμημάτων που ταίριαζαν σε οποιοδήποτε υπολογιστή. Σύντομα καταργήθηκε η ιδέα του «έτοιμου κουτιού», που αγόραζε ο χρήστης από μια εταιρεία, γιατί με λίγες γνώσεις ο καθένας μπορούσε να συναρμολογήσει το δικό του υπολογιστή.

Ο προσωπικός υπολογιστής απόκτησε μονάδα δισκέτας 3,5", σκληρό δίσκο και έγχρωμη οθόνη. Οι επεξεργαστές του έγιναν πιο γρήγοροι και εμφανίστηκαν τα πρώτα περιβάλλοντα γραφικών. Σήμερα ο προσωπικός υπολογιστής έχει βγει από το χώρο των φίλων της τεχνολογίας και έχει επικρατήσει σε επιχειρήσεις και υπηρεσίες όλων των ειδών. Πάρα πολλές εργασίες, που παλαιότερα απαιτούσαν ειδικά κατασκευασμένους υπολογιστές ή γίνονταν χειρωνακτικά, σήμερα γίνονται με απλά PC.

Η επιτυχία του προσωπικού υπολογιστή οφείλεται στην αρχιτεκτονική του, η οποία επιτρέπει τη «συναρμολόγηση» ενός υπολογιστή από τμήματα τα οποία απλώς ακολουθούν κάποιες προδιαγραφές λειτουργίας και επικοινωνίας. Πολλές μικρές εταιρείες μπόρεσαν να εξειδικευτούν κατασκευάζοντας μόνο μία κατηγορία τμημάτων για PC και έτσι να δημιουργηθεί μια μεγάλη αγορά υλικού. Αντίθετα, υπολογιστές άλλων κατηγοριών κατασκευάζονται εξ ολοκλήρου από μία μόνο εταιρεία, με συνέπεια την αύξηση του κόστους τους τόσο για την ίδια, όσο και για τον τελικό καταναλωτή.

Ο διάδρομος του προσωπικού υπολογιστή

Στον πρώτο προσωπικό υπολογιστή της IBM υπήρχε ένας κοινός επίπεδος διάδρομος πάνω στον οποίο συνδέονταν ο κεντρικός επεξεργαστής, η μνήμη και οι υπόλοιπες περιφερειακές μονάδες. Ο διάδρομος αυτός είχε εύρος 8 bit και δυνατότητα μετάδοσης 4,7 εκατομμυρίων bits το δευτερόλεπτο, με άλλα λόγια η ταχύτητα του ήταν 4,7 MHz. Στη συνέχεια, με την αύξηση της ταχύτητας των επεξεργαστών, επήλθε αντίστοιχη αύξηση τόσο στο εύρος του διαδρόμου όσο και στην ταχύτητά του.

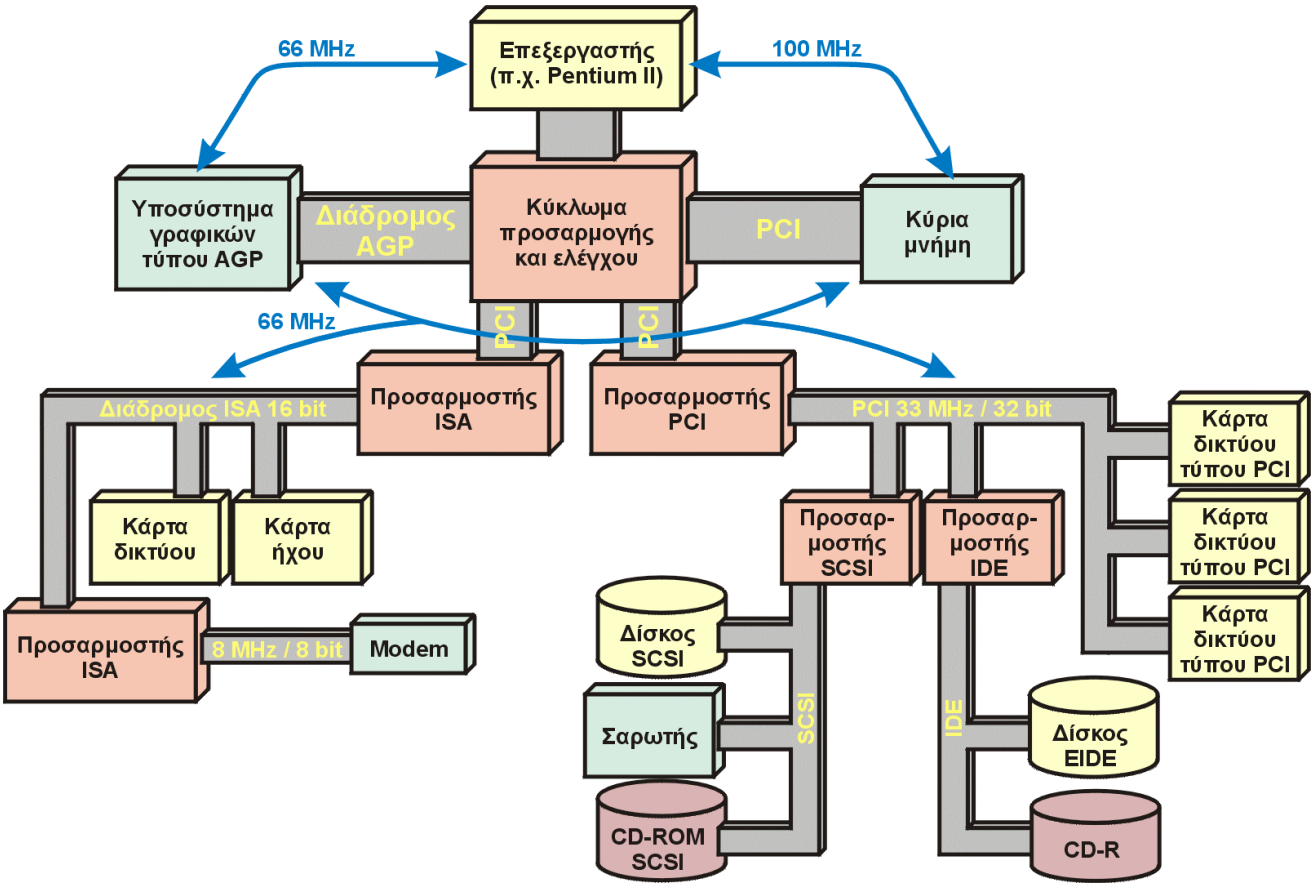
Ακολούθησε ο διάδρομος τύπου AT (**ISA**-Industry Standard Architecture) με εύρος 16 bit και ταχύτητες 6 και 8 MHz. Ο διάδρομος αυτός είχε συμβατότητα με περιφερειακά που απαιτούσαν εύρος 8 bits. Στη συνέχεια, με την εμφάνιση του επεξεργαστή 80386 της Intel, δημιουργήθηκε το πρότυπο **EISA** (Extended ISA) με εύρος 32 bits, διατηρώντας όμως την δυνατότητα σύνδεσης με περιφερειακές συσκευές παλαιότερου τύπου των 8 και 16 bits. Η IBM πρότεινε την αρχιτεκτονική **MCA** (MicroChannel Architecture) με αντίστοιχα χαρακτηριστικά (32 bits εύρος και ταχύτητα 10MHz), αλλά διαφορετικό τύπο διασύνδεσης· το πρότυπο αυτό τελικά απέτυχε.

Η επόμενη εξέλιξη στην τεχνολογία των διαδρόμων στους προσωπικούς υπολογιστές ήταν η εισαγωγή του τοπικού διαδρόμου ή **VL-Bus** (VESA Local Bus). Ο διάδρομος αυτός ανήκει στην κατηγορία διαδρόμου μνήμης-επεξεργαστή, αλλά επέτρεπε και την διασύνδεση έως και τριών συσκευών πάνω σε αυτόν. Η ταχύτητα λειτουργίας του ήταν 66 MHz και το εύρος 32 bits. Οι συσκευές που διασυνδέονταν πάνω στο VL-bus ήταν κυρίως το υποσύστημα γραφικών και ο ελεγκτής δίσκων. Το μεγαλύτερο μειονέκτημα του ήταν ότι λόγω του τρόπου ελέγχου του διαδρόμου από κάθε συσκευή, όσο περισσότερες συσκευές διασυνδέονταν, τόσο μειώνονταν η απόδοσή του. Έτσι οι περισσότερες υλοποιήσεις του VL-bus είχαν μέχρι 3 υποδοχές διασύνδεσης.

Το πρότυπο που τελικά επικράτησε, δίνοντας λύση στις μεγάλες ταχύτητες των επεξεργαστών και των μνημών και επιτρέποντας την διασύνδεσή και άλλων γρήγορων περιφερειακών, χωρίς τα μειονεκτήματα του VL-bus, ήταν το **PCI** (Peripheral Component Interconnect). Οι σύγχρονοι προσωπικοί υπολογιστές έχουν διάδρομο τύπου PCI με εύρος 32 ή 64 bit και ταχύτητες που φτάνουν τα 100MHz για να υποστηρίζουν επεξεργαστές των 350+ MHz.

Αρχικά, ο διάδρομος τύπου PCI είχε εύρος 32 bit και ταχύτητα 66MHz. Ανήκε στην κατηγορία των διαδρόμων μνήμης-επεξεργαστή, καθώς λίγες περιφερειακές συσκευές είχαν ανάγκη για την αυξημένη ταχύτητα που παρείχε το νέο αυτό πρότυπο. Έτσι οι μητρικές κάρτες των υπολογιστών είχαν, εκτός του διαδρόμου PCI και διάδρομο EISA ή ISA για διασύνδεση των υπολοίπων συσκευών. Καθώς όμως οι απαιτήσεις σε ταχύτητα των περιφερειακών συσκευών μεγάλωναν, ειδικά των καρτών δικτύου και των ελεγκτών σκληρών δίσκων, ο διάδρομος PCI έγινε επίπεδος διάδρομος.

Στις μέρες μας υπάρχουν πολλών ειδών κάρτες περιφερειακών για απευθείας διασύνδεση στο PCI. Η αύξηση στην ταχύτητα των επεξεργαστών στους προσωπικούς υπολογιστές επανέφερε ακόμα μια φορά το πρόβλημα της ύπαρξης ενός τοπικού διαδρόμου που να διασυνδέει τον επεξεργαστή, τη μνήμη και κάποιες εξίσου γρήγορες μονάδες. Η τεχνολογία απεικόνισης τρισδιάστατων γραφικών στους

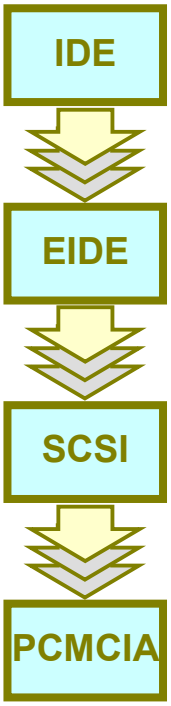


Αρχιτεκτονική σύγχρονου προσωπικού υπολογιστή

υπολογιστές και η μεγάλη υπολογιστική ισχύς που αυτή απαιτεί, δημιούργησε το πρότυπο διαδρόμου **AGP** (Accelerated Graphics Port). Ο διάδρομος AGP είναι ένας τοπικός διάδρομος που επιτρέπει την απευθείας επικοινωνία με τη μνήμη και τον επεξεργαστή με ταχύτητα 66MHz και εύρος 32 bits. Από την άλλη πλευρά, υπάρχει ένας διάδρομος PCI κατηγορίας μνήμης-επεξεργαστή που συνδέει τον επεξεργαστή, τη λανθάνουσα μνήμη και την κεντρική μνήμη με ταχύτητες 66 έως 100MHz και εύρος 32 bit. Επιπλέον υπάρχει ένας επίπεδος διάδρομος PCI πάνω στον οποίο διασυνδέονται τα υπόλοιπα περιφερειακά τύπου PCI, ενώ υπάρχουν και προσαρμοστές για τους διαδρόμους ISA, EISA, πάνω στους οποίους συνδέονται λιγότερο γρήγορες συσκευές.

Βλέπουμε λοιπόν ότι η οργάνωση των διαδρόμων ενός σύγχρονου προσωπικού υπολογιστή πλησιάζει σε πολυπλοκότητα και σε πλήθος επιπέδων αυτή ενός υπερυπολογιστή. Υπάρχουν διαφόρων κατηγοριών διάδρομοι, με διαφορετικά χαρακτηριστικά ώστε να προσαρμόζονται καλύτερα οι αντίστοιχες συσκευές και να γίνεται πλήρης εκμετάλλευση της ταχύτητας τους. Οι προσαρμοστικές μονάδες πλέον παίζουν καθοριστικό ρόλο στην συνολική απόδοση ενός υπολογιστή.

Διασύνδεση δίσκων



Στους προσωπικούς υπολογιστές οι σκληροί δίσκοι διασυνδέονται σε ένα ξεχωριστό διάδρομο που ονομάστηκε αρχικά διάδρομος **IDE** και κατόπιν **EIDE** (Enhanced IDE). Στο διάδρομο EIDE διασυνδέονται μέχρι και τέσσερις συσκευές, κυρίως δίσκοι και CD-ROM. Στην ουσία ο διάδρομος αποτελείται από δύο κανάλια, σε καθένα από τα οποία διασυνδέονται 2 συσκευές. Η μία από τις δύο προηγείται στον έλεγχο του καναλιού και ονομάζεται *κύρια* (master), ενώ η άλλη ονομάζεται *δευτερεύουσα* (slave). Η διάκριση αυτή επιτρέπει την συνύπαρξη αργών (CDROM) και γρήγορων (σκληροί δίσκοι) συσκευών στο ίδιο κανάλι. Η ταχύτητα του διαδρόμου αυτού φθάνει μέχρι και τα 20MB/sec. Το μεγαλύτερο πλεονέκτημα του προτύπου αυτού είναι η απλότητα της υλοποίησής του που επιτρέπει στους κατασκευαστές π.χ. σκληρών δίσκων να ενσωματώσουν πολλά από τα ηλεκτρονικά προσαρμογής IDE πάνω στο δίσκο και τελικά η μορφή του προσαρμοστή IDE στον διάδρομο PCI να είναι πολύ απλή.

Το πρότυπο διαδρόμου **SCSI** προέρχεται από τον χώρο των κύριων υπολογιστών και προσφέρει εξελιγμένα χαρακτηριστικά σε σχέση με το EIDE. Η μορφή αυτή του διαδρόμου επιτρέπει την διασύνδεση πολλών περιφερειακών μονάδων, κυρίως σκληρών δίσκων και μονάδων CDROM. Βέβαια είναι δυνατή η διασύνδεση και άλλων συσκευών που ακολουθούν το ίδιο πρότυπο, όπως σαρωτές, συσκευές εφεδρείας και άλλες.

Για την διασύνδεση του διαδρόμου SCSI με έναν διάδρομο Ε/Ε (π.χ. με PCI) χρειάζεται ένας προσαρμοστής SCSI. Στο διάδρομο αυτό μπορούν να συνδεθούν μέχρι και 7 συσκευές, η κάθε μία από τις οποίες έχει και ξεχωριστή διεύθυνση SCSI. Οι ταχύτητες SCSI ποικίλουν από 5MB/sec σε κανονική λειτουργία, 10MB/sec στο πρότυπο Fast SCSI, 20 MB/sec στο πρότυπο Ultra (Fast-Wide)-SCSI και 40MB/sec στο Ultra-Wide SCSI. Η ταχύτητα αυτή βέβαια είναι η μέγιστη δυνατή ταχύτητα διαδρόμου, και δεν αναφέρεται στην πραγματική ταχύτητα επικοινωνίας των περιφερειακών η οποία μπορεί να είναι αρκετά μικρότερη. Το πρότυπο SCSI είναι πολύ πιο αξιόπιστο και γρήγορο από το IDE· όμως η μεγάλη ποικιλία σε κατηγορίες και σε υλοποιήσεις, εμπόδιζε μέχρι πρόσφατα τη διάδοσή του στην κατηγορία των προσωπικών υπολογιστών. Στις μέρες μας υπάρχει πληθώρα περιφερειακών SCSI για προσωπικούς υπολογιστές συνυπάρχοντας μαζί με το πρότυπο διαδρόμου IDE.

Το **PCMCIA** είναι ένα νέο πρότυπο διαδρόμου για φορητούς υπολογιστές. Στον διάδρομο PCMCIA συνδέεται οποιοδήποτε είδος περιφερειακού το οποίο είναι σε μορφή κάρτας. Έτσι υπάρχουν κάρτες δικτύου, κάρτες μνήμης, κάρτες fax/modem ακόμα και δίσκοι IDE που έχουν μέγεθος αντίστοιχο μιας παχιάς πιστωτικής κάρτας. Οι κάρτες PCMCIA είναι ελαφρές, μικρές στο μέγεθος, συνδέονται και αποσυνδέονται εύκολα από το διάδρομο, δίνοντας την ευχέρεια σε έναν φορητό υπολογιστή να έχει ποικιλία περιφερειακών κατά περίπτωση, χωρίς να αυξάνεται το βάρος ή το μέγεθος του.

	ISA	EISA	MCA	VL	PCI
Εύρος	8-16 bit	32 bit	32 bit	32 bit	32-64 bit
Ρολόι	8 MHz	8,3 MHz	10 MHz	33 MHz	33,66-100 MHz
Διαμεταγωγή	16 MB/sec	32 MB/sec	40 MB/sec	132 MB/sec	>120 MB/sec
Περιφερειακά	>12	>12	>12	3	10

Plug and Play

Για κάθε περιφερειακή συσκευή υπάρχει ένα μικρό πρόγραμμα (ρουτίνα) το οποίο αναλαμβάνει το στοιχειώδη χειρισμό της κάθε φορά που ξεκινά να χρησιμοποιείται.

Τα προγράμματα αυτά μεταφέρονται από το BIOS σε συγκεκριμένες περιοχές της μνήμης RAM κατά την εκκίνηση του υπολογιστή και για το καθένα υπάρχει ένας μοναδικός αριθμός (*αριθμός διακοπής,* interrupt number) για το λειτουργικό σύστημα. Έτσι όταν μια περιφερειακή συσκευή στέλνει μια αίτηση για να εξυπηρετηθεί, αίτηση αυτή συνοδεύεται από τον αριθμό διακοπής, ώστε να διακοπεί προσωρινά το εκτελούμενο εκείνη τη στιγμή πρόγραμμα και να εκτελεστεί η απαραίτητη για τη συσκευή ρουτίνα.

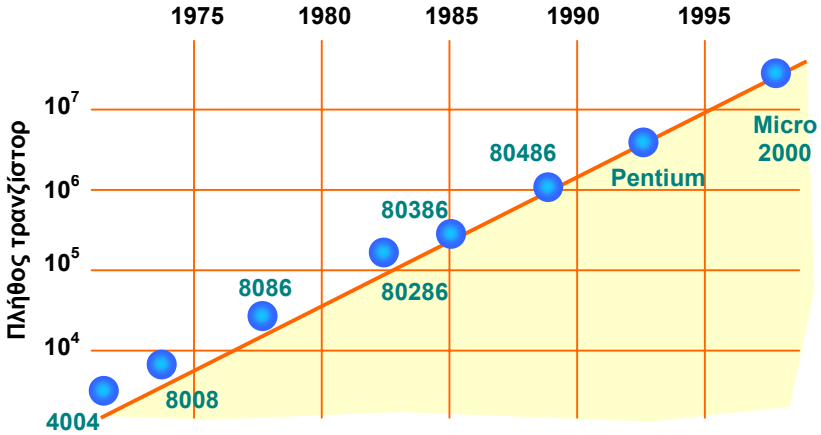
Ανάλογο μοναδικό για κάθε συσκευή σήμα (αριθμός) απαιτείται αν η εξυπηρέτηση πρόκειται να γίνει με DMA (μάθημα 5.1) ή αν πρέπει να δεσμευθεί μια θύρα Ε/Ε.

Οι υπολογιστές που δεν υποστηρίζουν το πρότυπο Plug and Play (ή για συντομία PnP) απαιτούν κυρίως από τον αριθμό διακοπής να είναι προκαθορισμένος σε κάθε περιφερειακή συσκευή, συνήθως με κάποιους *μικροδιακόπτες* (jumpers ή dip switches). Κάθε συσκευή μπορεί να λειτουργήσει το πολύ με δύο ή τρείς διαφορετικούς αριθμούς διακοπής και επιλέγεται κάποιος που θα είναι μοναδικός. Παρόλα αυτά υπάρχει ο κίνδυνος δύο συσκευές να έχουν προεπιλέξει τον ίδιο αριθμό διακοπής, οπότε δεν μπορούν να λειτουργήσουν μαζί.

Οι υπολογιστές που υποστηρίζουν το Plug and Play μπορούν να «διαπραγματευθούν» τους αριθμούς διακοπών κλπ. με τις περιφερειακές συσκευές, ώστε να μην υπάρχει πρόβλημα «σύγκρουσης» μεταξύ δύο συσκευών. Βέβαια και οι ίδιες οι συσκευές πρέπει να ακολουθούν το πρότυπο αυτό.

Επεξεργαστές

Ο πρώτος προσωπικός υπολογιστής, φτιαγμένος από την IBM, είχε τον επεξεργαστή 8088 της εταιρείας Intel. Στα χρόνια που ακολούθησαν η Intel κατασκεύαζε ολοένα ισχυρότερους επεξεργαστές (8086, 80286, 80386, …) που χρησιμοποιούνταν για τους προσωπικούς υπολογιστές. Όταν φάνηκε ότι το τμήμα της αγοράς των προσωπικών υπολογιστών ήταν πλέον πολύ μεγάλο και προσοδοφόρο, τότε και άλλες εταιρείες, όπως η AMD και η Cyrix, άρχισαν να κατασκευάζουν επεξεργαστές «συμβατούς» με αυτούς της Intel. Παρόλες τις δικαστικές προσπάθειες της Intel να διατηρήσει το μονοπώλιό της, επικαλούμενη παραβίαση των ευρεσιτεχνιών από τους ανταγωνιστές της, οι εταιρείες αυτές κατόρθωσαν να μοιραστούν την αγορά των επεξεργαστών. Όμως η Intel εξακολουθεί να διατηρεί ένα τεχνολογικό προβάδισμα δίνοντας κάθε φορά την κατεύθυνση στο σχεδιασμό νέων επεξεργαστών.



Ο Gordon Moore το 1965 έκανε την παρατήρηση ότι περίπου κάθε δύο χρόνια το πλήθος των τρανζίστορ στα ολοκληρωμένα κυκλώματα, η οποία επηρεάζει άμεσα την ισχύ των επεξεργαστών, θα διπλασιάζεται. Μέχρι σήμερα αυτή η πρόβλεψη έχει επαληθευθεί, και καλείται πλέον *νόμος του Moore*, που πρακτικά μας λέει ότι κάθε δύο χρόνια περίπου η ισχύς των προσωπικών υπολογιστών διπλασιάζεται.

Ο πιο κοινός επεξεργαστής σήμερα είναι ο Pentium της Intel και οι διάφοροι απόγονοί του (Pentium Pro, Pentium II, Celeron, Pentium III κλπ.)

Ο επεξεργαστής Pentium

Στις 22 Μαρτίου 1993 η Intel παρουσίασε τον Pentium, ένα νέο τότε επεξεργαστή που αποτελούσε μεγάλη σχεδιαστική βελτίωση σε σχέση με τους προκατόχους του. Περιέχοντας 3,1 εκατομμύρια τρανζίστορ σε ένα τετράγωνο ολοκληρωμένο κύκλωμα 5 × 5 εκ., πενταπλασίασε την ονομαστική υπολογιστική ισχύ τού αμέσως προηγούμενού του 80486.

Το ουσιαστικό αρχιτεκτονικό στοιχείο του Pentium, που προκάλεσε τη μεγάλη αυτή επιτάχυνση και το οποίο διατηρείται και στους νεότερους επεξεργαστές, είναι η *αρχιτεκτονική αγωγού* (pipeline), που παρουσιάστηκε στο μάθημα 3.5. Έτσι, ο Pentium περιέχει δύο αγωγούς εντολών, τους U και V, κάθε ένας από τους οποίους έχει πέντε στάδια. Αν δύο εντολές μπορούν να εκτελεστούν παράλληλα, τότε η μία ανατίθεται στον αγωγό U και η άλλη στον V, αλλιώς εκτελούνται η μία μετά την άλλη στον αγωγό U και ο V παραμένει ανενεργός.

Επίσης ο Pentium διαθέτει ενσωματωμένη *μονάδα πράξεων με αριθμούς κινητής υποδιαστολής* (Floating Point Unit, FPU), η οποία στα αρχικά στάδια των αντίστοιχων εντολών εκμεταλλεύεται τον αγωγό U και μετά προσθέτει άλλα τρία, σχηματίζοντας έτσι έναν αγωγό οκτώ σταδίων.

Περιφερειακή μνήμη

Όλα σχεδόν τα είδη περιφερειακής μνήμης είναι διαθέσιμα για τους προσωπικούς υπολογιστές, ορισμένα όμως είναι πολύ πιο κοινά και δημοφιλή.

Μαγνητικά μέσα

Ο *οδηγός δισκέτας* (floppy disk drive) συνοδεύει πιστά τον προσωπικό υπολογιστή από τη γέννησή του. Ξεκίνησε από τις 5,25" και χωρητικότητα 160 Kb, και τότε αποτελούσε τη μοναδική μονάδα περιφερειακής μνήμης. Η εκκίνηση του υπολογιστή γινόταν από δισκέτα και ο χρήστης έπρεπε να αλλάζει συχνά δισκέτες, αν είχε μόνο έναν οδηγό στη διάθεσή του.

Αργότερα παρουσιάστηκαν και οι οδηγοί των 3,5" με δισκέτες πιο μικρές και πιο ανθεκτικές. Σήμερα η χωρητικότητα των δισκετών φθάνει τα 2,88 MB (με πιο συνηθισμένα τα 1,44 MB) για τις 3,5" και τα 1,2 MB για τις 5,25", αν και οι δισκέτες των 5,25" είναι πλέον σπάνιες.

Ο πρώτος *σκληρός δίσκος* (hard disk) σε προσωπικό υπολογιστή εμφανίστηκε το 1983 και είχε την - τεράστια για την εποχή - χωρητικότητα των 10 MB. Παρέχοντας άφθονο χώρο αποθήκευσης για τις ανάγκες της εποχής, αξιοπιστία και ευκολία σε σχέση με τις δισκέτες, καθιερώθηκε αμέσως.

Σήμερα η χωρητικότητα των σκληρών δίσκων μετράται με GB, και η ταχύτητά τους έχει υπερδεκαπλασιαστεί.

Οι *μαγνητικές ταινίες* (magnetic tapes) στο παρελθόν χρησιμοποιούνταν σε μεγάλους υπολογιστές και υπερυπολογιστές, αρχικά ως μονάδες εισόδου και στη συνέχεια ως μονάδες διατήρησης *εφεδρικών αντιγράφων* (backup). Όταν έγιναν πιο μικρές και εύχρηστες, πέρασαν και στο χώρο των προσωπικών υπολογιστών για τη διατήρηση των εφεδρικών αντιγράφων.

Μία μαγνητική ταινία έχει χωρητικότητα από 100 MB έως αρκετά GB και χρειάζεται περίπου 2-4 ώρες για να γραφτεί ολόκληρη. Είναι λοιπόν ιδανική για εφεδρικά αντίγραφα.

Οπτικά μέσα

Μία συνηθισμένη περιφερειακή συσκευή που λειτουργεί με οπτικές μεθόδους, δηλαδή με ακτίνες laser, είναι ο *οδηγός CD-ROM* (CD-ROM drive). Ο οδηγός αυτός διαβάζει οπτικούς δίσκους (CD) με δεδομένα ή μουσική. Οι δίσκοι CD-ROM έχουν χωρητικότητα 650 MB ή 74 λεπτά μουσικής.

Όταν πρωτοεμφανίστηκαν, οι οδηγοί CD-ROM είχαν ρυθμό μεταφοράς δεδομένων της τάξης των KB/s και χρόνους προσπέλασης πάνω από 200 ms. Σήμερα έχουμε οδηγούς 32 ή και 40 φορές πιο γρήγορους, με χρόνους προσπέλασης που είναι συγκρίσιμοι με σκληρούς δίσκους και ρυθμούς μεταφοράς δεδομένων αρκετά MB/s.

Όλα σχεδόν τα προγράμματα διανέμονται από τις εταιρείες λογισμικού σε δίσκους CD-ROM, οι οποίοι χωρούν συνήθως όλα αρχεία του προγράμματος – ενώ θα χρειάζονταν πολλές δισκέτες για την ίδια δουλειά.

Οι οδηγοί CD-ROM μπορούν μόνο να διαβάσουν από τους δίσκους, αλλά υπάρχουν και *οδηγοί εγγραφής και ανάγνωσης* (CD-R). Με τον περιορισμό βέβαια ότι κάθε CD-ROM μπορεί να εγγραφεί μόνο μία φορά, οι οδηγοί αυτοί δίνουν τη δυνατότητα στους χρήστες να εγγράψουν CD με μουσική, να κρατούν εφεδρικά αντίγραφα των αρχείων τους ή να μοιράζονται αρχεία μεγάλου μεγέθους.

Μεγαλύτερης χωρητικότητας είναι οι *δίσκοι DVD* (Digital Versatile Disk), οι οποίοι είναι επίσης μόνο ανάγνωσης. Οι δίσκοι αυτοί έχουν χωρητικότητες που φθάνουν τα 17 GB, και έτσι είναι κατάλληλοι για ήχο υψηλής ποιότητας, ταινίες μεγάλης διάρκειας κλπ.

Ανακεφαλαίωση

Από το 1981 που εμφανίστηκε μέχρι σήμερα, ο προσωπικός υπολογιστής έχει γίνει ο πιο δημοφιλής υπολογιστής για οικιακή και επαγγελματική χρήση.

Ο πρώτος προσωπικός υπολογιστής είχε επίπεδο διάδρομο τύπου ISA. Στη συνέχεια δημιουργήθηκαν ταχύτερα πρότυπα όπως το VL-Bus και το PCI για να καλύψουν τις αυξημένες ανάγκες σε ταχύτητα του επεξεργαστή, της μνήμης, του υποσυστήματος γραφικών κ.ά.

Οι πιο πολλοί προσωπικοί υπολογιστές χρησιμοποιούν επεξεργαστές της εταιρίας Intel ή παρόμοιους, που σήμερα έχουν αρχιτεκτονική αγωγού. Υπάρχουν όλα τα είδη περιφερειακής μνήμης για προσωπικούς υπολογιστές, και πιο δημοφιλή είναι τα μαγνητικά – σκληροί δίσκοι, δισκέτες, μαγνητικές ταινίες – και τα οπτικά (CD-ROM, DVD).

Γλωσσάριο όρων

Αριθμός Διακοπής	*Interrupt Number*
Αρχιτεκτονική Αγωγού	*Pipeline Architecture*
Δευτερεύουσα Συσκευή	*Slave Device*
Γέφυρα PCI	*PCI Bridge*
Διάδρομος	*Bus*
Δίσκος DVD	*Digital Versatile Disk*
Εφεδρικό Αντίγραφο	*Backup*
Κύρια Συσκευή	*Master Device*
Μαγνητική Ταινία	*Magnetic Tape*
Μικροδιακόπτης	*Jumper ή Dip Switch*
Μονάδα Πράξεων με Αριθμούς Κινητής Υποδιαστολής	*Floating Point Unit – FPU*
Οδηγός CD-ROM	*CD-ROM Drive*
Οδηγός Δισκέτας	*Floppy Disk Drive*
Προσωπικός Υπολογιστής	*Personal Computer - PC*
Σκληρός Δίσκος	*Hard Disk*
Τοπικός Διάδρομος	*Local Bus*

Ερωτήσεις

? Πόσο μεγάλη είναι η εξέλιξη στους προσωπικούς υπολογιστές από την εμφάνισή τους μέχρι σήμερα;

? Σε τι οφείλουν την επιτυχία τους οι προσωπικοί υπολογιστές;

? Ποιος είναι ο ρόλος του διαδρόμου στο PC;

? Ποια πρόταση αναφέρουμε ως «νόμο του Moore»;

? Περίγραψε την αρχιτεκτονική των επεξεργαστών της οικογένειας Pentium.

? Ποιες είναι οι κυριότερες συσκευές περιφερειακής μνήμης σε ένα PC;

Μάθημα 6.2 Η Μητρική Κάρτα του Προσωπικού Υπολογιστή

Σκοπός του μαθήματος

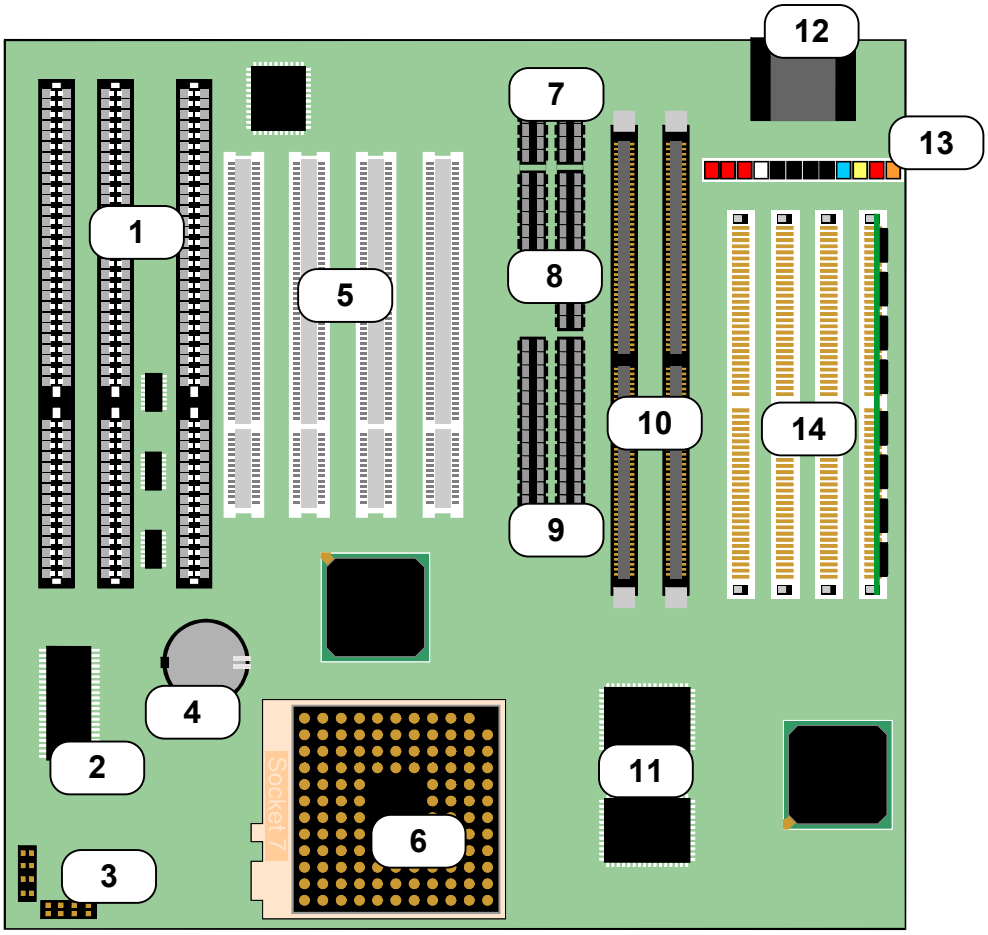
Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τα στοιχεία που απαρτίζουν τη μητρική κάρτα ενός προσωπικού υπολογιστή.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να υποδεικνύεις σε μια μητρική κάρτα τα κυριότερα στοιχεία της.*
- *Να περιγράφεις τη βασική τους λειτουργικότητα*

Η *μητρική κάρτα* (motherboard) είναι ένα μεγάλο τυπωμένο κύκλωμα, το οποίο υλοποιεί τη βασική συνδεσμολογία ενός προσωπικού υπολογιστή. Έχει υποδοχές για



τις διάφορες μονάδες και μερικά πολύ βασικά ολοκληρωμένα κυκλώματα.

Στο σχήμα της προηγούμενης σελίδας βλέπουμε τη διάταξη των διαφόρων μονάδων πάνω σε μία μητρική κάρτα. Ο σχεδιασμός αυτός δεν είναι ίδιος για όλες τις κάρτες, αλλά η μορφή των μονάδων και των υποδοχών είναι όμοια.

Οι μονάδες που σημειώνονται στο σχήμα είναι:

1 *Υποδοχές επέκτασης* (expansion slots) τύπου ISA για περιφερειακές συσκευές «παλαιάς» τεχνολογίας των 8 και 16 bits.

2 Το ολοκληρωμένο κύκλωμα για το *βασικό σύστημα εισόδου-εξόδου* (Basic Input-Output System, BIOS) του υπολογιστή. Το BIOS αποτελείται από μερικά βασικά προγράμματα, τα οποία έχουν γραφτεί από τον κατασκευαστή της μητρικής κάρτας και παρέχουν τις βασικές υπηρεσίες πρόσβασης στο υλικό του υπολογιστή. Τα λειτουργικά συστήματα χρησιμοποιούν το υλικό μέσω των υπηρεσιών αυτών. Έτσι δε χρειάζεται κάθε λειτουργικό σύστημα να λαμβάνει υπόψη του τις μικροϊδιαιτερότητες κάθε τύπου μητρικής κάρτας, γιατί αυτές «καλύπτονται» από το BIOS. Το ολοκληρωμένο κύκλωμα του BIOS είναι μια μνήμη *μόνο ανάγνωσης* (Read-Only Memory, ROM).

Ορισμένα από τα χαρακτηριστικά λειτουργίας του υπολογιστή (όπως π.χ. η ημερομηνία και η ώρα του ρολογιού του υπολογιστή, ή οι μονάδες δισκέτας που περιέχει ο υπολογιστής) μπορούν να μεταβληθούν. Για το λόγο αυτό, οι τιμές τους κρατούνται σε μία μικρή μνήμη του BIOS και ονομάζονται *ρυθμίσεις του BIOS (BIOS settings).*

3 *Μικροδιακόπτες ρυθμίσεων* (jumpers). Αυτοί αποτελούνται από σειρές ακίδων, οι οποίες μπορούν να είναι συνδεδεμένες μεταξύ τους ή όχι. Ανάλογα με το πώς είναι συνδεδεμένες οι ακίδες, η μητρική κάρτα λειτουργεί με διάφορες ρυθμίσεις, οι οποίες αφορούν π.χ. την ταχύτητα του ρολογιού συστήματος, το μέγεθος της λανθάνουσας μνήμης L1, την τάση λειτουργίας του επεξεργαστή κλπ. Μικροδιακόπτες ρυθμίσεων βρίσκονται σε διάφορα σημεία της μητρικής κάρτας.

4 Μπαταρία λιθίου για τη λειτουργία του ρολογιού πραγματικού χρόνου του υπολογιστή και για τη διατήρηση των ρυθμίσεων του BIOS που γίνονται με λογισμικό.

5 Υποδοχές επέκτασης τύπου PCI για κάρτες επέκτασης των 32 bits. Για τις περισσότερες περιφερειακές συσκευές (π.χ. την κάρτα δικτύου) υπάρχουν κάρτες τύπου PCI αλλά και παλαιότερες κάρτες τύπου ISA.

6 Η υποδοχή τοποθέτησης του επεξεργαστή. Όπως βλέπουμε στο σχήμα, η υποδοχή αυτή έχει θέσεις για ακίδες σε όλη την επιφάνειά της. Αυτό γίνεται, γιατί οι σύγχρονοι επεξεργαστές έχουν πάρα πολλές ακίδες επικοινωνίας, και η περιφέρεια του ολοκληρωμένου κυκλώματος δεν επαρκεί για να τοποθετηθούν. Έτσι πολλές από τις ακίδες τοποθετούνται στην κάτω επιφάνεια του επεξεργαστή.

Όταν ο επεξεργαστής είναι τοποθετημένος στην υποδοχή του, δε φαίνεται όπως τα υπόλοιπα ολοκληρωμένα κυκλώματα. Πάνω από τον επεξεργαστή τοποθετείται η *ψύκτρα* (cooler fan), ένας μικρός ανεμιστήρας, ο οποίος απομακρύνει τη θερμότητα που εκλύεται λόγω της λειτουργίας του ολοκληρωμένου κυκλώματος για να λειτουργεί αυτό σωστά. Στους σύγχρονους επεξεργαστές, που έχουν πολύ μεγάλη

πυκνότητα κυκλωμάτων ανά τετραγωνικό εκατοστό, η καλή λειτουργία της ψύκτρας είναι απαραίτητη, γιατί υπερθερμαίνονται πολύ εύκολα.

7 Ελεγκτής της *σειριακής θύρας* (serial port) του υπολογιστή. Στις υποδοχές αυτές συνδέονται οι έξοδοι που συνήθως βλέπουμε στο πίσω μέρος του υπολογιστή. Οι έξοδοι σειριακής επικοινωνίας χρησιμοποιούνται κατά κύριο λόγο για τη σύνδεση του ποντικιού και modem.

8 Ελεγκτής των οδηγών δισκέτας και της *παράλληλης θύρας* (parallel port). Η δεύτερη υποδοχή συνήθως οδηγεί σε μια έξοδο στο πίσω μέρος του υπολογιστή, όπου συνδέουμε εκτυπωτές και άλλες περιφερειακές συσκευές που υποστηρίζουν παράλληλη επικοινωνία.

9 Ελεγκτής IDE για σκληρούς δίσκους, CD-ROM κλπ. Κάθε μητρική κάρτα έχει δύο τέτοιους ελεγκτές και σε κάθε ένα μπορούν να συνδεθούν το πολύ δύο συσκευές με ένα καλώδιο που έχει δύο υποδοχές. Η μία από τις δύο συσκευές είναι η *κύρια* συσκευή του ελεγκτή (master) ενώ η άλλη είναι η *δευτερεύουσα* (slave). Η επιλογή της συσκευής που θα είναι κύρια για τον κάθε ελεγκτή γίνεται με μικροδιακόπτες ρυθμίσεων των ίδιων των συσκευών.

! Αν και οι δύο συσκευές που είναι συνδεδεμένες στον ίδιο ελεγκτή είναι ρυθμισμένες για να λειτουργούν ως κύριες, τότε ο υπολογιστής δεν μπορεί να ξεκινήσει.

10 Υποδοχές σύνδεσης για κυκλώματα μνήμης τύπου DIMM που έχουν 168 ακίδες. Τα κυκλώματα αυτά θα τα δούμε στο επόμενο μάθημα.

11 Υποδοχές σύνδεσης και κυκλώματα λανθάνουσας μνήμης L2, η οποία είναι συνήθως στατική μνήμη.

12 Έξοδος σύνδεσης του πληκτρολογίου πάνω στη μητρική κάρτα. Η έξοδος αυτή δε μεταφέρει μόνο τα δεδομένα από το πληκτρολόγιο, αλλά και το ηλεκτρικό ρεύμα για τη λειτουργία του.

13 Σύνδεση τροφοδοσίας ηλεκτρικού ρεύματος από το τροφοδοτικό. Από το σημείο αυτό ξεκινούν καλώδια τα οποία παρέχουν ρεύμα στις διάφορες περιφερειακές συσκευές που συνδέουμε στον υπολογιστή, τις ψήκτρες κλπ. Επίσης από το σημείο αυτό παρέχεται μέσω των τυπωμένων κυκλωμάτων της μητρικής κάρτας ρεύμα στα υπόλοιπα ολοκληρωμένα κυκλώματα και τις υποδοχές της.

14 Υποδοχές σύνδεσης για κυκλώματα μνήμης τύπου SIMM που έχουν 72 ακίδες. Στο σχήμα φαίνεται ένα τέτοιο κύκλωμα τοποθετημένο στη δεξιότερη υποδοχή. Συνήθως όμως, όταν ο επεξεργαστής είναι τύπου Pentium ή νεότερος, τα κυκλώματα τύπου SIMM τοποθετούνται ανά ζεύγη. Για τα κυκλώματα αυτά θα πούμε περισσότερα στο επόμενο μάθημα.

Ανακεφαλαίωση

Στη μητρική κάρτα του προσωπικού υπολογιστή υπάρχουν θέσεις για τη σύνδεση όλων των μονάδων που τον απαρτίζουν. Οι κυριότερες τέτοιες θέσεις είναι για:
- Τον επεξεργαστή
- Τα κυκλώματα της μνήμης
- Τους ελεγκτές των μονάδων περιφερειακής μνήμης
- Διάφορες κάρτες επέκτασης
- Συσκευές εισόδου / εξόδου δεδομένων.

Γλωσσάριο όρων

Βασικό Σύστημα Εισόδου-Εξόδου	*Basic Input-Output System - BIOS*
Δευτερεύουσα Συσκευή	*Slave Device*
Κύρια Συσκευή	*Master Device*
Μητρική Κάρτα	*Motherboard*
Μνήμη Μόνο Ανάγνωσης	*Read-Only Memory - ROM*
Παράλληλη Θύρα	*Parallel Port*
Ρυθμίσεις του BIOS	*BIOS Settings*
Σειριακή Θύρα	*Serial Port*
Υποδοχή Επέκτασης	*Expansion Slot*
Ψύκτρα	*Cooler Fan*

Ερωτήσεις

? Ποιες είναι οι βασικότερες υποδοχές σε μια μητρική κάρτα;

? Πώς τροφοδοτούνται με ηλεκτρικό ρεύμα οι περιφερειακές συσκευές και οι κάρτες επέκτασης σε ένα προσωπικό υπολογιστή;

? Πού συνδέεται το ποντίκι σε έναν προσωπικό υπολογιστή;

? Πού συνδέεται το πληκτρολόγιο σε έναν προσωπικό υπολογιστή;

? Ποιες συσκευές συνδέουμε στη σειριακή θύρα;

? Ποιες συσκευές συνδέουμε στην παράλληλη θύρα;

Μάθημα 6.3 Η Μνήμη στον Προσωπικό Υπολογιστή

Σκοπός του μαθήματος

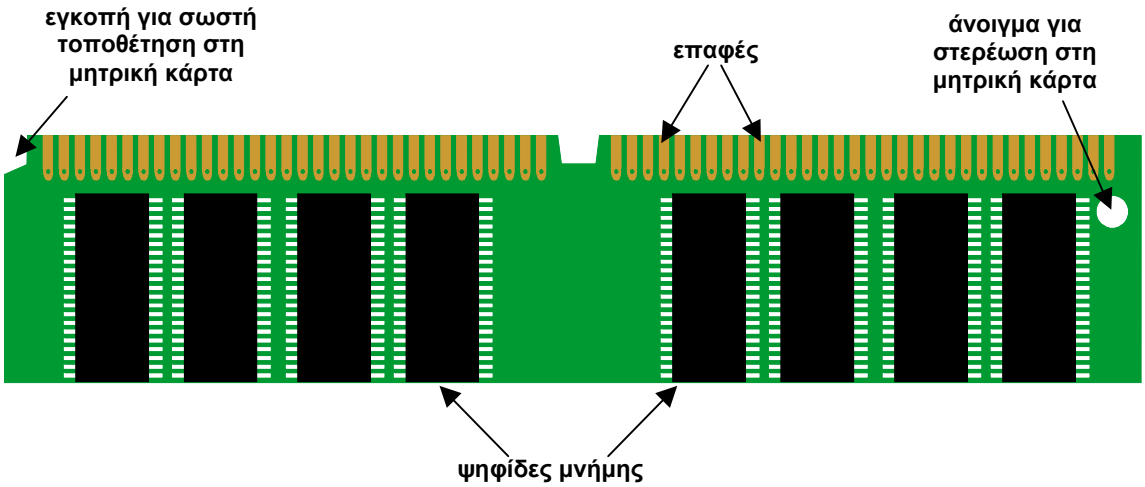
Σκοπός του μαθήματος αυτού είναι να περιγράψει τα διάφορα είδη μνήμης και των αντίστοιχων ολοκληρωμένων κυκλωμάτων που χρησιμοποιούνται στους προσωπικούς υπολογιστές.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να αναγνωρίζεις και κατατάσσεις τα ολοκληρωμένα κυκλώματα μνήμης των προσωπικών υπολογιστών.*
- *Να εξηγείς τις διαφορές μεταξύ των στατικών και των δυναμικών μνημών.*
- *Να απαριθμείς και περιγράφεις τους τύπους των δυναμικών μνημών.*
- *Να διακρίνεις τα διάφορα είδη λανθάνουσας μνήμης στον προσωπικό υπολογιστή.*
- *Να περιγράφεις την ιεραρχία της μνήμης στον προσωπικό υπολογιστή.*

Στον προσωπικό υπολογιστή η *κύρια μνήμη* (main memory) είναι οργανωμένη σε ολοκληρωμένα κυκλώματα των 4, 8, 16 ή 32 Mbytes. Η μητρική κάρτα έχει υποδοχές πάνω στις οποίες συνδέονται τα κυκλώματα αυτά.



Ένα τέτοιο κύκλωμα μνήμης βλέπουμε στο σχήμα. Στη μία πλευρά το ολοκληρωμένο κύκλωμα έχει μεταλλικές επαφές, οι οποίες συνδέονται με τις υποδοχές της μητρικής κάρτας. Στη μία ή και τις δύο επιφάνειες του κυκλώματος βρίσκονται ψηφίδες μνήμης με την ίδια χωρητικότητα, η οποία μετριέται σε Mbit. Στο σχήμα μας οι ψηφίδες είναι 8· αν καθε μία από αυτές έχει χωρητικότητα 4 Mbit, τότε όλο το κύκλωμα μνήμης έχει χωρητικότητα 8×4 Mbit = 32 Mbit, δηλαδή 4 MB.

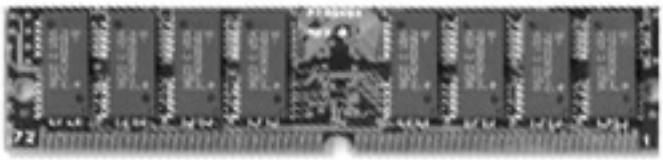
Μερικές από τις επαφές του κυκλώματος μνήμης χρησιμεύουν για τη μεταφορά της διεύθυνσης μνήμης προς το κύκλωμα, ενώ άλλες κάνουν τη μεταφορά των δεδομένων από το κύκλωμα (όταν διαβάζεται η μνήμη) ή προς αυτό (όταν εγγράφεται). Υπάρχουν βέβαια και επαφές που ενεργοποιούν την ανάγνωση και την εγγραφή. Για να διαβαστεί ή να εγγραφεί μία θέση της μνήμης, οι επαφές της διεύθυνσης ενεργοποιούνται με τέτοιο τρόπο ώστε να μεταδοθεί στο κύκλωμα η κατάλληλη διεύθυνση καθώς και το κατάλληλο σήμα. Μετά από ένα χρονικό διάστημα, οι επαφές δεδομένων ενεργοποιούνται με το περιεχόμενο της θέσης μνήμης που ζητήθηκε – στην ανάγνωση – ή απλώς ολοκληρώνεται η εγγραφή.

Ο χρόνος που περνά για να ολοκληρωθεί μια λειτουργία από το κύκλωμα μνήμης ονομάζεται *χρόνος προσπέλασης* (access time) της μνήμης και μετράται σε ns (10^{-9} s).

SIMM και DIMM

Όπως είδαμε στο προηγούμενο μάθημα, οι μητρικές κάρτες έχουν συνήθως δύο είδη υποδοχών για κυκλώματα μνήμης. Το ένα είδος υποδοχών δέχεται κυκλώματα μνήμης τύπου SIMM και το άλλο δέχεται κυκλώματα τύπου DIMM.

Τα κυκλώματα τύπου *SIMM* (Single In-line Memory Module) έχουν 72 επαφές και ταιριάζουν στις πιο μικρές υποδοχές μνήμης της μητρικής κάρτας. Σε κάθε λειτουργία μπορούν να μεταδοθούν 4 bytes από ή προς τα κυκλώματα αυτά. Τα κυκλώματα SIMM τροφοδοτούνται με ηλεκτρικό ρεύμα τάσης 5V.



Τα κυκλώματα τύπου *DIMM* (Dual In-line Memory Module) έχουν 168 επαφές και ταιριάζουν στις μεγάλες υποδοχές μνήμης της μητρικής πλακέτας. Επειδή έχουν περισσότερες επαφές, μεταφέρουν 8 bytes δεδομένων σε κάθε λειτουργία. Τα κυκλώματα DIMM τροφοδοτούνται με ηλεκτρικό ρεύμα τάσης 5V ή 3,3V.

Μπορούμε να χρησιμοποιήσουμε στον ίδιο υπολογιστή και κυκλώματα SIMM και κυκλώματα DIMM, αρκεί τα κυκλώματα DIMM να λειτουργούν με ρεύμα 5V. Αν χρησιμοποιήσουμε DIMM που λειτουργούν στα 3,3V μαζί με κυκλώματα SIMM, υπάρχει σοβαρός κίνδυνος καταστροφής τους.

Οι πιο σύχρονοι επεξεργαστές της εταιρίας Intel, συγκεκριμένα οι Pentium και οι νεότεροι, μεταφέρουν 8 bytes δεδομένων από και προς τη μνήμη κάθε φορά. Έτσι, σε τέτοιους υπολογιστές συνήθως χρησιμοποιούμε κυκλώματα DIMM είτε κυκλώματα SIMM σε ζεύγη.

Μνήμες DRAM, EDO, SDRAM, SRAM

Με την εξέλιξη των προσωπικών υπολογιστών, έχουν εμφανιστεί πολλοί τύποι κυκλωμάτων μνήμης. Κάθε ένας από αυτούς προσφέρει μια βελτίωση σε σχέση με τους παλαιότερους και όλοι έχουν σαν στόχο να κάνουν την πρόσβαση στη μνήμη όσο πιο γρήγορη γίνεται.

Οι δύο βασικές κατηγορίες μνημών είναι οι *δυναμικές* (dynamic) και οι *στατικές* (static). Οι δυναμικές μνήμες χρησιμοποιούνται γενικά για την κύρια μνήμη (RAM), ενώ οι στατικές βρίσκουν τη χρήση τους στη *λανθάνουσα μνήμη* (cache) του υπολογιστή.

Δυναμικές μνήμες

DRAM

Ο πιο συνηθισμένος τύπος κυκλωμάτων μνήμης είναι οι μνήμες *DRAM* (Dynamic RAM). Στις μνήμες αυτές η τιμή του κάθε bit αποθηκεύεται σε ένα μικροσκοπικό πυκνωτή. Οι πυκνωτές αυτοί όμως αποφορτίζονται - και μάλιστα πολύ σύντομα, περίπου σε 1 ms - έτσι το περιεχόμενό τους κινδυνεύει να χαθεί. Για να μη γίνει αυτό, ειδικά κυκλώματα επαναφορτίζουν περιοδικά τους πυκνωτές. Το όνομα της μνήμης, «δυναμική», προέρχεται από αυτή τη συνεχή διαδικασία της επαναφόρτισης.

Οι μνήμες DRAM ξεκίνησαν από χρόνους πρόσβασης της τάξης των 120 ns και με την εξέλιξη της τεχνολογίας έφτασαν τα 60 ns. Όμως, αν σκεφτούμε πόσες λειτουργίες μνήμης μπορούν να γίνουν σε 1 δευτερόλεπτο με ένα τέτοιο χρόνο πρόσβασης (1 s / 60 ns $\cong 17 \cdot 10^6$ λειτουργίες, δηλαδή συχνότητα λειτουργιών 10 MHz) και τις συγκρίνουμε με τις ταχύτητες των επεξεργαστών, βλέπουμε ότι οι μνήμες αυτές είναι πολύ αργές. Στο πρόβλημα αυτό δόθηκαν δυο λύσεις: η μία ήταν η προσθήκη λανθάνουσας μνήμης (cache) και η άλλη ήταν η κατασκευή πιο γρήγορων τύπων μνημών.

EDO

Πιο γρήγορες είναι οι μνήμες *EDO* (extended-data-out), οι οποίες είναι ένας τύπος DRAM. Οι μνήμες αυτές απελευθερώνουν τα τμήματά τους που δέχονται τη διεύθυνση και τα δεδομένα εισόδου νωρίτερα από τις κοινές DRAM, και έτσι μπορούν να προχωρήσουν στην επόμενη λειτουργία τους πιο γρήγορα, επιταχύνοντας τις προσβάσεις στη μνήμη έως 40%.

Όλοι οι τύποι μνήμης που αναφέραμε, εκτός από μερικές DRAM πολύ παλαιάς τεχνολογίας που δεν χρησιμοποιούνται πλέον, σε κάθε λειτουργία ανάγνωσης μεταφέρουν 4 bytes. Έτσι κάθε φορά εκτός από τη διεύθυνση που ζητήθηκε (που περιέχει 1 byte), διαβάζουν και τις γειτονικές θέσεις μνήμης. Πολύ συχνά οι γειτονικές θέσεις μνήμης διαβάζονται διαδοχικά και δεν απαιτείται για όλες μια πλήρης λειτουργία ανάγνωσης. Έτσι η πρόσβαση στη μνήμη επιταχύνεται.

SDRAM

Καλύτερες επιδόσεις αλλά με μεγαλύτερο κόστος επιτυγχάνουν οι μνήμες *SDRAM* (Synchronous DRAM), οι οποίες, σε αντίθεση με τις υπόλοιπες δυναμικές μνήμες που έχουμε αναφέρει ως τώρα, λειτουργούν συγχρονισμένα με το ρολόι του συστήματος.

Οι μνήμες SDRAM βρίσκονται μόνο σε μορφή DIMM και σε κάθε ανάγνωση μεταφέρουν 8 bytes.

Στατικές μνήμες

SRAM

Σε αντίθεση με τις δυναμικές μνήμες, οι στατικές μνήμες *SRAM* (static RAM) δε χρειάζονται ανανέωση για να μη χαθούν τα περιεχόμενά τους. Αυτό τις κάνει πολύ πιο γρήγορες, αλλά συγχρόνως και πολύ πιο ακριβές. Έτσι βρίσκουν κυρίως χρήση στη λανθάνουσα μνήμη των προσωπικών υπολογιστών, η οποία έχει μικρό μέγεθος, μικρότερο από 1 Mb.

Οι στατικές μνήμες μπορούν να λειτουργούν ασύγχρονα σε σχέση με το ρολόι του υπολογιστή, οπότε αναφέρονται ως Async SRAM ή σε συγχρονισμό με αυτό, οπότε αναφέρονται ως Sync SRAM.

Λανθάνουσα μνήμη στον προσωπικό υπολογιστή

Η λανθάνουσα μνήμη, όπως είδαμε στο μάθημα 4.4, βοηθά στην επιτάχυνση της λειτουργίας ενός επεξεργαστή κρατώντας το περιεχόμενο των πιο πρόσφατα χρησιμοποιημένων θέσεων μνήμης.

Οι προσωπικοί υπολογιστές διαθέτουν και αυτοί λανθάνουσα μνήμη. Μάλιστα, εξαιτίας της διαφοράς στην ταχύτητα των επεξεργαστών και των κυκλωμάτων κύριας μνήμης, οι πιο πολλοί προσωπικοί υπολογιστές έχουν δύο διαφορετικούς τύπους λανθάνουσας μνήμης, που είναι γνωστοί σαν λανθάνουσα μνήμη «επιπέδου 1» (Level 1 ή L1 cache) και «επιπέδου 2» (Level 2 ή L2 cache).

Λανθάνουσα μνήμη L1

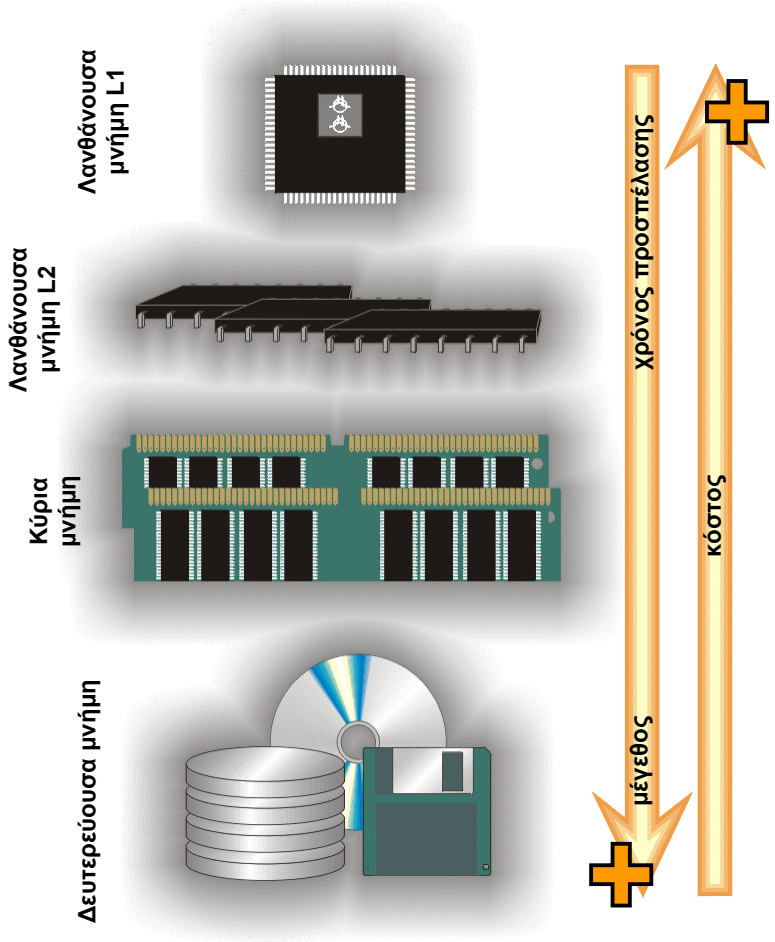
Η λανθάνουσα μνήμη επιπέδου 1 δεν είναι άλλη από αυτή που βρίσκεται μέσα στον ίδιο τον επεξεργαστή. Ο πρώτος επεξεργαστής που περιείχε λανθάνουσα μνήμη ήταν ο 80486 της Intel, με 8 Kb. Στη συνέχεια όλοι οι επεξεργαστές των προσωπικών υπολογιστών περιείχαν λανθάνουσα μνήμη με μεγέθη που φτάνουν τα 32 Kb. Εσωτερικά η λανθάνουσα μνήμη L1 χωρίζεται σε ομάδες των 16 ή 32 bytes.

Η λανθάνουσα μνήμη L1 κρατά διευθύνσεις μνήμης που αντιστοιχούν σε δεδομένα, αλλά και σε εντολές μηχανής. Συχνά χωρίζεται σε δύο τμήματα για τα δύο αυτά είδη διευθύνσεων. Οι εντολές μηχανής που εκτελούνται εσωτερικά στον επεξεργαστή είναι ιδιαίτερα χρήσιμες, όταν ο επεξεργαστής έχει αρχιτεκτονική αγωγού (pipeline), που επεξεργάζεται πολλές εντολές συγχρόνως.

Λανθάνουσα μνήμη L2

Η λανθάνουσα μνήμη επιπέδου 2 είναι πιο μεγάλη σε μέγεθος από την L1, αλλά όχι τόσο γρήγορη, και βρίσκεται πάνω στη μητρική κάρτα του υπολογιστή. Όπως είπαμε και πιο πριν, τα κυκλώματά της κατά κύριο λόγο αποτελούνται από στατική μνήμη. Η λανθάνουσα μνήμη επιπέδου 2 έχει μέγεθος συνήθως που μπορεί να φθάσει και το 1 Mb, αλλά το μέγιστό της μέγεθος εξαρτάται και από τη μητρική κάρτα.

Η ιεραρχία μνήμης λοιπόν στον προσωπικό υπολογιστή φαίνεται στο σχήμα:



ROM

ROM

Σε κάθε μητρική κάρτα υπάρχει και η μνήμη ROM, η οποία περιέχει το βασικό σύστημα εισόδου-εξόδου του υπολογιστή (BIOS). Η μνήμη αυτή μπορεί να διαβαστεί μόνο και όχι να γραφτεί. Στην πραγματικότητα η μνήμη ROM είναι πολύ διαφορετική από τις μνήμες RAM, παρόλο που έχει επικρατήσει να αναφέρονται μαζί ως κύρια μνήμη. Πρόκειται για ένα λογικό κύκλωμα ρυθμισμένο έτσι, ώστε για κάθε αριθμητική τιμή εισόδου (τη διεύθυνση) να δίνει μία ορισμένη τιμή εξόδου (το περιεχόμενο της διεύθυνσης).

Ανακεφαλαίωση

Η κύρια μνήμη στον προσωπικό υπολογιστή είναι οργανωμένη σε κυκλώματα τύπου SIMM ή DIMM. Υπάρχουν διάφορα είδη τέτοιων κυκλωμάτων, τα οποία διαφέρουν μεταξύ τους ως προς την ταχύτητα πρόσβασης των δεδομένων στη μνήμη.

Στον προσωπικό υπολογιστή υπάρχει λανθάνουσα μνήμη σε δύο επίπεδα: το πρώτο επίπεδο είναι πιο γρήγορο και μικρό σε μέγεθος και βρίσκεται μέσα στον επεξεργαστή, ενώ το δεύτερο βρίσκεται πάνω στη μητρική κάρτα.

Γλωσσάριο όρων

Δυναμική Μνήμη	*Dynamic Memory*
Κυκλώματα Τύπου DIMM	*Dual In-line Memory Module*
Κυκλώματα Τύπου SIMM	*Single In-line Memory Module*
Κύρια Μνήμη	*Main Memory*
Λανθάνουσα Μνήμη	*Cache*
Λανθάνουσα Μνήμη «Επιπέδου 1»	*Level 1 - L1 Cache*
Λανθάνουσα Μνήμη «Επιπέδου 2»	*Level 2 - L2 Cache*
Μνήμες DRAM	*Dynamic RAM*
Μνήμες EDO	*Extended-Data-Out*
Μνήμες SDRAM	*Synchronous DRAM*
Μνήμες SRAM	*Static RAM*
Στατική Μνήμη	*Static Memory*
Χρόνος Προσπέλασης	*Access Time*

Ερωτήσεις

? Ποιοι είναι οι τύποι ολοκληρωμένων κυκλωμάτων μνήμης του προσωπικού υπολογιστή;

? Ποια είναι τα χαρακτηριστικά των μνημών τύπου SIMM;

? Ποια είναι τα χαρακτηριστικά των μνημών τύπου DIMM;

? Ποια είναι η διαφορά μεταξύ στατικών και δυναμικών μνημών;

? Ποιοι είναι σήμερα οι τύποι δυναμικών μνημών;

? Πού χρησιμοποιούμε κυρίως τις στατικές μνήμες;

? Ποια είναι η ιεραρχία της μνήμης στον προσωπικό υπολογιστή;

Τι μάθαμε σε αυτό το κεφάλαιο

- Η αρχιτεκτονική του Προσωπικού Υπολογιστή είναι οργανωμένη γύρω από το διάδρομό του, που έχει φτάσει σε εξελιγμένα πρότυπα και ικανότητες.
- Οι πιο πολλοί προσωπικοί υπολογιστές χρησιμοποιούν επεξεργαστές της εταιρίας Intel ή παρόμοιους.
- Στη μητρική κάρτα του προσωπικού υπολογιστή υπάρχουν θέσεις για τη σύνδεση όλων των μονάδων που τον απαρτίζουν.
- Η κύρια μνήμη στον προσωπικό υπολογιστή είναι οργανωμένη σε κυκλώματα τύπου SIMM ή DIMM.
- Στους νεότερους προσωπικούς υπολογιστές υπάρχει λανθάνουσα μνήμη δύο επιπέδων.

Βιβλιογραφία – Πηγές

- Maran R. Μαθαίνω μόνος μου τους υπολογιστές και το Internet, Γεννάδειος Σχολή, 1995.
- Δημόπουλου Κ., Α. Παρασκευόπουλου, 80x86-Αρχιτεκτονική-Σχεδίαση-Προγραμματισμός, Παπασωτηρίου,1992.
- Πώς δουλεύουν οι Υπολογιστές, Κλειδάριθμος, 1994.
- Alexandridis N., Design of Microprocessor - Based System, Prentice Hall, 1993.
- Hutchinson S., Sawyer S., Computers and Information Systems, Irwin, 1996.
- Vermaat M., Walker T., Hall T., Repede J., Discovering Computer, Shelly Cashman Series, 1997.

Κεφάλαιο

Λειτουργικά Συστήματα - Διεργασίες

Σκοπός του κεφαλαίου αυτού είναι να παρουσιάσει τα Λειτουργικά Συστήματα, και να περιγράψει τις πιο βασικές οντότητες του Λειτουργικού Συστήματος, τις διεργασίες.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να εξηγείς τι είναι Λειτουργικό Σύστημα και ποιες είναι οι αρμοδιότητές του.
- Να περιγράφεις το ρόλο της διεργασίας σε ένα Λειτουργικό Σύστημα.
- Να ορίζεις το πρόβλημα του κρίσιμου τμήματος και να προτείνεις τρόπους για τη λύση του.

Μαθήματα

7.1 Λειτουργικά Συστήματα

7.2 Διεργασίες και Ελαφρές Διεργασίες

7.3 Απεικόνιση Διεργασιών

7.4 Κρίσιμα Τμήματα και Αμοιβαίος Αποκλεισμός

7.5 Σηματοφορείς

Μάθημα 7.1 Λειτουργικά Συστήματα

Σκοπός του μαθήματος

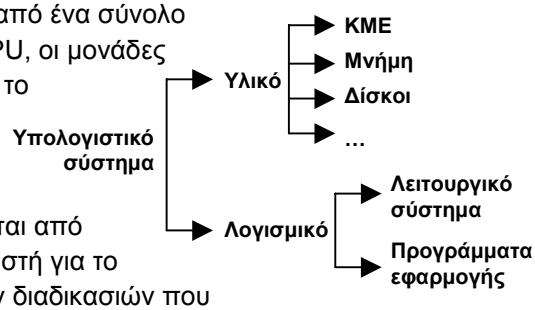
Σκοπός του μαθήματος αυτού είναι να ορίσει τα Λειτουργικά Συστήματα, να παρουσιάσει την ιστορία τους και να περιγράψει τη βασική τους δομή.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τι είναι Λειτουργικό Σύστημα*
- *Να απαριθμείς τις διάφορες κατηγορίες Λειτουργικών Συστημάτων που έχουν εμφανιστεί*
- *Να περιγράφεις τη βασική δομή ενός Λειτουργικού Συστήματος*

Όπως γνωρίζουμε, κάθε υπολογιστικό σύστημα αποτελείται (α) από ένα σύνολο συσκευών, (όπως η Κεντρική Μονάδα Επεξεργασίας - ΚΜΕ / CPU, οι μονάδες αποθήκευσης όπως μαγνητικοί και οπτικοί δίσκοι, οι εκτυπωτές, το πληκτρολόγιο κλπ), οι οποίες ονομάζονται *υλικό* του υπολογιστή (hardware), και (β) από το *λογισμικό* (software) το οποίο αποτελείται από το *Λειτουργικό Σύστημα* και τα *προγράμματα εφαρμογής*. Τα προγράμματα εφαρμογής γράφονται από τους χρήστες-προγραμματιστές και δίνουν εντολές στον υπολογιστή για το πώς θα χρησιμοποιήσει τις συσκευές για την εκτέλεση διαφόρων διαδικασιών που συνδέονται με το υπολογιστικό σύστημα.



> Το Λειτουργικό Σύστημα (ΛΣ) είναι ένα σύνολο προγραμμάτων που λειτουργεί ως σύνδεσμος ανάμεσα στα προγράμματα του χρήστη και το υλικό και καθορίζει τον τρόπο λειτουργίας του υπολογιστικού συστήματος, ελέγχοντας και συντονίζοντας τη χρήση των μονάδων του από τα διάφορα προγράμματα εφαρμογής των χρηστών.

> Αν φανταστούμε τον υπολογιστή ως ένα ταξί με τους χρήστες και τα προγράμματά τους να αποτελούν τους επιβάτες, τότε το ΛΣ παίζει το ρόλο του οδηγού, χωρίς τη συνεχή παρουσία του οποίου το ταξί είναι άχρηστο

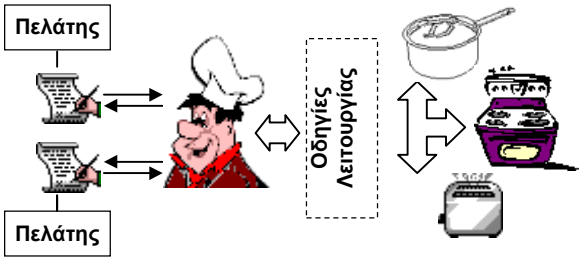
Οι στόχοι ενός ΛΣ είναι:

1. **Η διευκόλυνση των χρηστών**. Τα ΛΣ υπάρχουν επειδή κάνουν πιο εύκολη τη χρήση των υπολογιστικών συστημάτων και δίνουν τη δυνατότητα σε ανθρώπους με μικρές γνώσεις γύρω από τους υπολογιστές να εκτελέσουν πολύπλοκες εργασίες.

2. **Η διευκόλυνση των προγραμματιστών**. Χωρίς ΛΣ κάθε πρόγραμμα έπρεπε π.χ. να ελέγχει τακτικά το πληκτρολόγιο για είσοδο από το χρήστη, να γνωρίζει τις ακριβείς εντολές που πρέπει να στείλει στον εκτυπωτή για να τυπώσει κάτι ή να οργανώνει μόνο του το χώρο αποθήκευσης των δεδομένων του σε ένα σκληρό δίσκο.

3. **Η αποδοτική λειτουργία του υπολογιστικού συστήματος**, δηλαδή η όσο το δυνατόν καλύτερη χρησιμοποίηση του υλικού, ώστε να κατανέμεται καλύτερα το υπολογιστικό φορτίο. Το ΛΣ διαθέτει τη «γενική εικόνα» όλων των προγραμμάτων που πρέπει να εκτελεστούν, όλων των χρηστών του υπολογιστικού συστήματος και των αναγκών τους· έτσι, μπορεί να ρυθμίσει καλύτερα πότε και ποια προγράμματα θα εκτελεστούν κλπ.

Ο ρόλος του ΛΣ μπορεί να περιγραφεί με μια παρομοίωση από την καθημερινή ζωή, η οποία θα χρησιμοποιείται στη συνέχεια, όπου χρειάζεται, για τη διευκρίνιση διαφόρων εννοιών. Ας παρομοιώσουμε λοιπόν ένα υπολογιστικό σύστημα με ένα ζαχαροπλαστείο το οποίο παρασκευάζει γλυκά κατά παραγγελία και σύμφωνα με τις οδηγίες που δίνουν οι πελάτες. Ο υπάλληλος του ζαχαροπλαστείου παίζει το ρόλο της Κεντρικής Μονάδας Επεξεργασίας (ΚΜΕ, CPU), και τα σκεύη του είναι οι υπόλοιπες συσκευές του υπολογιστικού συστήματος. Οι οδηγίες για την παρασκευή των γλυκών είναι τα προγράμματα και οι πελάτες είναι οι χρήστες του υπολογιστικού συστήματος. Οι οδηγίες που ακολουθεί ο υπάλληλος του ζαχαροπλαστείου για την εξυπηρέτηση των πελατών αντιστοιχούν στο ΛΣ του υπολογιστικού συστήματος.



Λειτουργικά Συστήματα Ομαδικής Επεξεργασίας

Ένας πρώτος τρόπος λειτουργίας του ζαχαροπλαστείου είναι οι πελάτες να αναμένουν σε μια ουρά και να δίνουν με τη σειρά στον ζαχαροπλάστη τις οδηγίες τους για την παρασκευή των γλυκών της προτίμησής τους. Μόνο όταν ολοκληρωθεί η εργασία για ένα πελάτη εξυπηρετείται ο επόμενος. Μια καλύτερη λύση είναι να ομαδοποιούνται πελάτες που θέλουν το ίδιο γλυκό και να εξυπηρετούνται διαδοχικά ώστε να γίνεται καλύτερη χρήση των σκευών.

Αυτός ο τρόπος εξυπηρέτησης αντιστοιχεί στα ΛΣ πρώτης γενιάς της δεκαετίας του ’50, τα *Λειτουργικά Συστήματα Ομαδικής Επεξεργασίας* (batch processing). Στα συστήματα αυτά ο χειριστής του υπολογιστικού συστήματος ομαδοποιούσε τα προγράμματα που υπέβαλλαν οι χρήστες· έτσι το ΛΣ δεχόταν μια ομάδα ομοειδών προγραμμάτων (π.χ. προγραμμάτων COBOL), τα επεξεργαζόταν το ένα μετά το άλλο (με τη σειρά εμφανίσεως) και τύπωνε τα αποτελέσματά τους πάλι με την ίδια σειρά. Τα πρώτα αυτά λειτουργικά συστήματα, παρουσίαζαν τα ακόλουθα βασικά μειονεκτήματα:

♦ **Υποαπασχόληση των συσκευών**: Σε ένα ΛΣ ομαδικής επεξεργασίας δεν ήταν δυνατή η ταυτόχρονη εκτέλεση περισσοτέρων του ενός προγραμμάτων. Έπρεπε πρώτα να τελειώσει ένα πρόγραμμα, για να αρχίσει η εκτέλεση του επομένου προγράμματος της ομάδας. Ένα πρόγραμμα όμως δεν μπορεί να απασχολεί ταυτόχρονα όλες τις μονάδες του Η/Υ, οι οποίες κατά συνέπεια υποαπασχολούνταν· για παράδειγμα, η ΚΜΕ απασχολούνταν συνήθως σε ποσοστό μικρότερο του 10%.

Εάν ήταν δυνατό να εκτελούνται πολλά προγράμματα ταυτόχρονα, θα μπορούσε σε μια δεδομένη χρονική στιγμή κάθε ένα από αυτά να απασχολεί και διαφορετική μονάδα. Έτσι θα περιοριζόταν ο ανενεργός χρόνος κάποιων μονάδων του υπολογιστή.

♦ Το χρονικό διάστημα που μεσολαβεί από τη στιγμή που ο προγραμματιστής θα αφήσει στον υπολογιστή το πρόγραμμά του μέχρι τη στιγμή που θα πάρει τα αποτελέσματα, ονομάζεται *χρόνος ανακύκλωσης* (turnaround time) και στα ΛΣ ομαδικής επεξεργασίας το διάστημα αυτό ήταν πολύ μεγάλο. Συνήθως τα αποτελέσματα των προγραμμάτων δεν ήταν διαθέσιμα, παρά μόνο αφού είχε ολοκληρωθεί η εκτέλεση όλης της ομάδας. Έτσι ο χρόνος ανακύκλωσης εξαρτιόταν από το χρόνο εκτέλεσης ολόκληρης της ομάδας, ο οποίος μπορεί να είναι π.χ. 50 ή 100 φορές μεγαλύτερος από το χρόνο εκτέλεσης του συγκεκριμένου προγράμματος.

Λειτουργικά Συστήματα Πολυπρογραμματισμού

> Η εξυπηρέτηση των πελατών του ζαχαροπλαστείου θα ήταν πιο γρήγορη αν ο υπάλληλος είχε οδηγίες από τον ιδιοκτήτη του ζαχαροπλαστείου να παρασκευάζει περισσότερα από ένα γλυκά ταυτοχρόνως. Κατά τη διάρκεια π.χ. του ψησίματος ενός γλυκού έχει τη δυνατότητα να προετοιμάζει στον αναμίκτη (μίξερ) ένα άλλο, εκμεταλλευόμενος το χρόνο του. Όταν όμως ολοκληρωθεί το ψήσιμο του πρώτου θα τον ειδοποιεί π.χ. ένα κουδουνάκι να διακόψει την προετοιμασία του δευτέρου και να συνεχίσει την παρασκευή του πρώτου γλυκού με το επόμενο στάδιό της.

Αυτή την ιδέα οργάνωσης ακολουθούν τα ΛΣ δεύτερης γενιάς (δεκαετίας ’60), τα *Λειτουργικά Συστήματα Πολυπρογραμματισμού* (multiprogramming), στα οποία επιδιώκεται η μείωση του άεργου χρόνου των μονάδων του υπολογιστή και του χρόνου ανακύκλωσης. Η διακοπή μιας λειτουργίας (το κουδουνάκι του ζαχαροπλαστείου) γίνεται με τη δημιουργία ειδικών σημάτων, των *σημάτων διακοπής* (interrupts), τα οποία διακόπτουν την τρέχουσα λειτουργία της ΚΜΕ.

Πολλές φορές το ΛΣ Πολυπρογραμματισμού ορίζεται ως το ΛΣ που επιτρέπει σε περισσότερα από ένα προγράμματα να είναι «φορτωμένα» στη μνήμη τού υπολογιστή και να εκτελούνται συγχρόνως. Φυσικά μόνο ένα πρόγραμμα μπορεί να απασχολεί ανά πάσα στιγμή κάθε μονάδα· π.χ. ένα πρόγραμμα εκτελείται από την ΚΜΕ, ένα άλλο διαβάζει από το δίσκο, ένα τρίτο στέλνει δεδομένα στη σειριακή θύρα εξόδου κλπ.

> Μια πολύ πρακτική υπηρεσία για τους πελάτες του ζαχαροπλαστείου είναι οι τηλεφωνικές παραγγελίες και η κατ’ οίκον διανομή των γλυκών, οι οποίες αντιστοιχούν στην τηλεπεξεργασία που θα δούμε αμέσως μετά.

Παράλληλα με τα ΛΣ Πολυπρογραμματισμού αναπτύχθηκαν και συσκευές οι οποίες έδιναν τη δυνατότητα σε περιφερειακά όπως τηλέτυπα, οθόνες, εκτυπωτές κλπ. να επικοινωνήσουν με ένα υπολογιστή μέσω τηλεπικοινωνιακών γραμμών. Έτσι οι περιφερειακές μονάδες δεν ήταν πλέον απαραίτητο να βρίσκονται στον ίδιο φυσικό χώρο με τον υπολογιστή. Τα ΛΣ που διαχειρίζονταν τέτοια υπολογιστικά συστήματα με «απομακρυσμένα» περιφερειακά ονομάζονται *Λειτουργικά Συστήματα Τηλεπεξεργασίας*. Παρ’ όλα τα πλεονεκτήματα που παρέχουν τα ΛΣ Πολυπρογραμματισμού, παρουσιάζουν και μειονεκτήματα. Ένα απ’ αυτά είναι π.χ. ότι το ίδιο το ΛΣ είναι πλέον ένα πολύπλοκο πρόγραμμα (σε αντίθεση με τα ΛΣ ομαδικής επεξεργασίας που είναι πολύ απλά), το οποίο με τη σειρά του απασχολεί τον

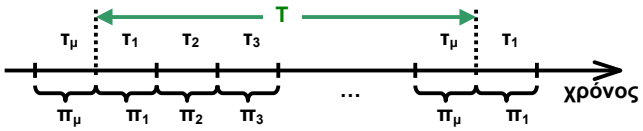
υπολογιστή σε βάρος των προγραμμάτων εφαρμογών, και μάλιστα τόσο πιο πολύ, όσο πιο πολύπλοκο είναι.

Λειτουργικά Συστήματα Καταμερισμού Χρόνου

Τερματικό
Συνδυασμός οθόνης και πληκτρολογίου για κάθε χρήστη ενός μεγάλου υπολογιστή

Πολλά μεγάλα υπολογιστικά συστήματα εξυπηρετούν πολλούς χρήστες, κάθε ένας από τους οποίους έχει στη διάθεσή του ένα τερματικό. Ο χρήστης πληκτρολογεί εντολές στο τερματικό, μέσω του οποίου διαβιβάζονται και εκτελούνται στο υπολογιστικό σύστημα. Για να έχουν όλοι οι χρήστες την εντύπωση ότι εξυπηρετούνται παράλληλα, το υπολογιστικό σύστημα καταμερίζει το χρόνο του, δίνοντας από λίγο και εκ περιτροπής στον καθένα.

> Μια άλλη μέθοδος λειτουργίας του ζαχαροπλαστείου θα ήταν η «ταυτόχρονη» παρασκευή των γλυκών, με καταμερισμό του χρόνου του υπαλλήλου μεταξύ των διαφόρων συνταγών. Κάθε πέντε λεπτά χτυπά ένα ρολόι και ο ζαχαροπλάστης διακόπτει την εργασία του, βάζει στην άκρη το γλυκό που έφτιαχνε και καταπιάνεται με το επόμενο στη σειρά.



Στα υπολογιστικά συστήματα το ρόλο του ρολογιού έχει ένα ειδικό κύκλωμα, ο *χρονιστής* (timer), το οποίο σε τακτά χρονικά διαστήματα δημιουργεί ένα σήμα διακοπής. Ο χρόνος χωρίζεται σε διαστήματα διάρκειας T. Αν υπάρχουν μ προγράμματα για εκτέλεση, που συμβολίζονται με $\pi_1,\pi_2,\ldots,\pi_\mu$, η περίοδος T διαιρείται σε μ χρονικά διαστήματα $\tau_1,\tau_2,\ldots,\tau_\mu$. Ο χρονιστής δημιουργεί ένα σήμα στην αρχή κάθε περιόδου τ_ν, οπότε αρχίζει να εξυπηρετείται από την ΚΜΕ το ν-οστό πρόγραμμα. Αν κάποιο από αυτά δεν ζητά εξυπηρέτηση ή έχει τελειώσει, η σειρά του δίνεται στο επόμενο.

Ένα ΛΣ που οργανώνει τη λειτουργία του υπολογιστή κατ' αυτό τον τρόπο ονομάζεται *Λειτουργικό Σύστημα Καταμερισμού Χρόνου* (time sharing).

Η περίοδος T μπορεί να είναι της τάξης των 3 sec· αν υποθέσουμε ότι υπάρχουν 6 προγράμματα για εκτέλεση, στο καθένα (σε κάθε περίοδο) διατίθεται 0,5 sec, χρόνος αρκετά μεγάλος για να καλύψει πολλές φορές ολόκληρη την εκτέλεση ενός προγράμματος.

Εάν κάποιο πρόγραμμα δεν καλύπτεται, περιμένει μέχρι την επόμενη χρονική περίοδο. Μερικές φορές βέβαια ένα πρόγραμμα πρέπει να εκτελεστεί αμέσως, χωρίς καθυστέρηση, επειδή η ταχύτητα απόκρισής του έχει ουσιαστική σημασία. Τότε, σε περιπτώσεις δηλαδή λειτουργίας σε *πραγματικό χρόνο* (real-time operation), το πρόγραμμα έχει προτεραιότητα έναντι των άλλων και εξυπηρετείται πρώτο.

Λειτουργικά Συστήματα 3ης και 4ης γενιάς

Σε πολλά νεότερα ΛΣ, αυτά της *3ης γενιάς*, γίνεται συνδυασμός των παραπάνω μεθόδων, για να υποστηρίζουν ταυτόχρονα ομαδική επεξεργασία και καταμερισμό χρόνου. Η έννοια της ομαδικής επεξεργασίας έχει σήμερα διαφοροποιηθεί και καθορίζεται βασικά από την έλλειψη αλληλεπίδρασης μεταξύ του χρήστη και του προγράμματός του.

Η ανάπτυξη νέων αρχιτεκτονικών για το υλικό, όπως π.χ. παράλληλων υπολογιστών, κατανεμημένων συστημάτων καθώς και δικτύων υπολογιστών, έδωσε την αφορμή για την ανάπτυξη ακόμα πιο εξελιγμένων και πολύπλοκων ΛΣ, αυτών της *4ης γενιάς*. Στα ΛΣ αυτά γίνεται πραγματικά ταυτόχρονη εκτέλεση πολλών προγραμμάτων, αφού κάθε επεξεργαστής μπορεί να ασχολείται με ένα διαφορετικό πρόγραμμα.

Τα συστήματα πολυεπεξεργαστών αντιστοιχούν στην ύπαρξη πολλών ζαχαροπλαστών που μοιράζονται τα ίδια σκεύη· τα κατανεμημένα συστήματα αντιστοιχούν σε πολλούς υπαλλήλους, καθένα με τα δικά του σκεύη και τέλος τα δίκτυα υπολογιστών είναι αντίστοιχα με μια αλυσίδα συνεργαζομένων ζαχαροπλαστείων.

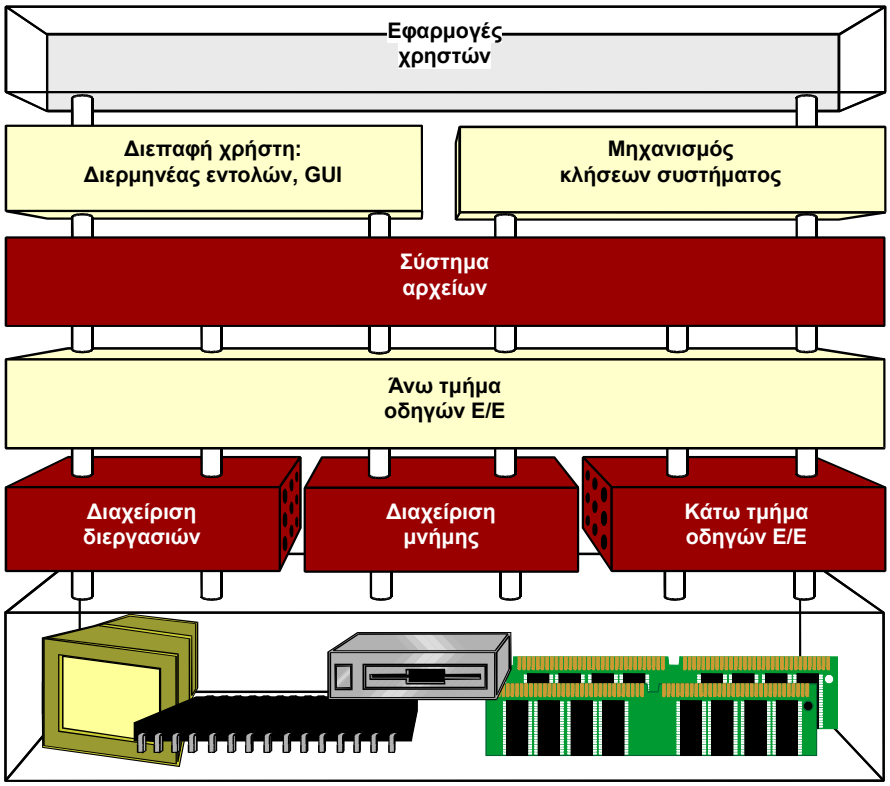
Στα υπολογιστικά συστήματα, το πρόγραμμα του ΛΣ συνήθως πωλείται από τον κατασκευαστή μαζί με το υλικό και είναι καθοριστικής σημασίας για τη λειτουργία του. Είναι δυνατόν ο ίδιος υπολογιστής να χρησιμοποιεί κατά καιρούς διαφορετικά ΛΣ, ανάλογα με τις ανάγκες των χρηστών του, και να έχει τελείως διαφορετική συμπεριφορά και απόδοση. Ακόμα, είναι δυνατόν υπολογιστές με διαφορετικό υλικό να χρησιμοποιούν το ίδιο ΛΣ και να έχουν παρόμοια συμπεριφορά, πιθανώς με διαφορετική απόδοση.

Τα ΛΣ είναι γραμμένα σε συμβολική γλώσσα ή σε συνδυασμό συμβολικής γλώσσας και κάποιας γλώσσας υψηλού επιπέδου (όπως η C).

Δομή ενός Λειτουργικού Συστήματος

Τα περισσότερα ΛΣ, και ιδιαίτερα τα σύγχρονα, είναι οργανωμένα σε *επίπεδα* (layers). Αυτό σημαίνει ότι κατά τη σχεδίασή τους έχουν διαιρεθεί σε τμήματα, και κάθε τμήμα τους επικοινωνεί μόνο με αυτά που βρίσκονται στο αμέσως ανώτερο ή το αμέσως κατώτερο επίπεδο. Όσα τμήματα χρησιμοποιούν απευθείας το υλικό του υπολογιστή, βρίσκονται στο κατώτερο επίπεδο του ΛΣ. Τα υπόλοιπα τμήματα, που βρίσκονται σε ανώτερα επίπεδα, δεν επικοινωνούν καθόλου με το υλικό, αλλά χρησιμοποιούν τα τμήματα που ανήκουν στο αμέσως κατώτερο επίπεδο.

Στο σχήμα φαίνεται ένα παράδειγμα οργάνωσης ΛΣ σε επίπεδα. Η οργάνωση αυτή βέβαια είναι ενδεικτική, γιατί υπάρχουν πολλές παραλλαγές της, αλλά η βασική φιλοσοφία είναι κοινή.



Στο χαμηλότερο επίπεδο ανήκουν τα τμήματα του ΛΣ που διαχειρίζονται:

- ☑ τη μνήμη
- ☑ τα υπό εκτέλεση προγράμματα
- ☑ τις λειτουργίες επικοινωνίας με τις περιφερειακές συσκευές.

Οι λειτουργίες αυτές αποτελούν το κάτω τμήμα των *οδηγών συσκευών Ε/Ε* (device drivers), ειδικών τμημάτων του ΛΣ που αναλαμβάνουν την επικοινωνία του ΛΣ με τα περιφερειακά. Στο αμέσως ανώτερο επίπεδο βρίσκεται το άνω τμήμα

των οδηγών Ε/Ε και πάνω από αυτό βρίσκεται το επίπεδο που κάνει τη διαχείριση του συστήματος αρχείων. Τα προγράμματα των χρηστών επικοινωνούν μόνο με το υψηλότερο επίπεδο σε ένα ΛΣ, που αποτελείται από τη *διεπαφή χρήστη* (user interface) και τις κλήσεις συστήματος. Η διεπαφή με το χρήστη μπορεί να γίνεται είτε με εντολές, με το *διερμηνέα εντολών* (command interpreter), ή με μία *διεπαφή χρήστη με χρήση γραφικών* (Graphical User Interface, GUI).

Παρατηρήστε ότι όπως είδαμε γενικά για τον υπολογιστή στο μάθημα 1.1, και το λειτουργικό σύστημα έχει ιεραρχική οργάνωση.

Ανακεφαλαίωση

Το Λειτουργικό Σύστημα είναι ένα σύνολο προγραμμάτων που οργανώνουν και επιβλέπουν τις λειτουργίες του υλικού σε ένα υπολογιστικό σύστημα. Κατά την εξέλιξή τους έχουν εμφανιστεί διάφορες κατηγορίες ΛΣ, όπως τα ΛΣ ομαδικής επεξεργασίας (1η γενιά), πολυπρογραμματισμού και καταμερισμού χρόνου (2η γενιά), πολυεπεξεργασίας (3η και 4η γενιά) κλπ. Τα περισσότερα ΛΣ είναι οργανωμένα σε επίπεδα, με το κατώτερο επίπεδο μόνο να επικοινωνεί απευθείας με το υλικό του υπολογιστή, και τα προγράμματα του χρήστη να χρησιμοποιούν μόνο το ανώτερο επίπεδο.

Γλωσσάριο όρων

Διεπαφή Χρήστη	*User Interface*
Διεπαφή Χρήστη Με Χρήση Γραφικών	*Graphical User Interface - GUI*
Διερμηνέας Εντολών	*Command Interpreter*
Επίπεδο	*Layer*
Καταμερισμός Χρόνου	*Time Sharing*
Κεντρική Μονάδα Επεξεργασίας – ΚΜΕ	*Central Processing Unit - CPU*
Λειτουργία Πραγματικού Χρόνου	*Real-Time Operation*
Λειτουργικό Σύστημα	*Operating System*
Λογισμικό	*Software*
Οδηγός Συσκευής	*Device Driver*
Ομαδική Επεξεργασία	*Batch Processing*
Πολυπρογραμματισμός	*Multiprogramming*
Πρόγραμμα Εφαρμογής	*Application Program*
Σήμα Διακοπής	*Interrupt*
Τηλεπεξεργασία	*Remote Processing*
Υλικό	*Hardware*
Χρονιστής	*Timer*
Χρόνος Ανακύκλωσης	*Turnaround Time*

Ερωτήσεις

? Ποια είναι τα βασικά τμήματα ενός υπολογιστικού συστήματος;

? Ποιο είναι το έργο ενός Λειτουργικού Συστήματος; Τι θα γινόταν αν δεν υπήρχαν ΛΣ;

? Πόσες γενιές ΛΣ έχουν εμφανιστεί, και τι είδους ΛΣ αποτελούν την κάθε γενιά;

? Περίγραψε την οργάνωση ενός ΛΣ σε επίπεδα. Τι πλεονεκτήματα νομίζεις ότι προσφέρει η οργάνωση αυτή;

Μάθημα 7.2 Διεργασίες και Ελαφρές Διεργασίες

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει την έννοια της διεργασίας, που είναι πολύ βασική στα ΛΣ, την έννοια της ελαφριάς διεργασίας και να δείξει τις ομοιότητες και τις διαφορές τους.

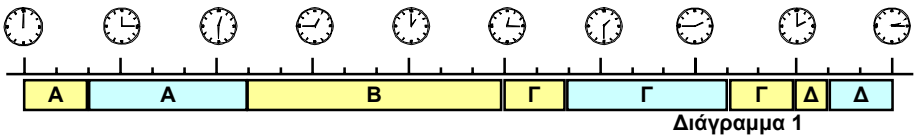
Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να περιγράφεις πώς ένα ΛΣ μπορεί να κατανέμει το χρόνο του μεταξύ πολλών προγραμμάτων*
- *Να εξηγείς τι είναι η διεργασία και σε τι διαφέρει από ένα πρόγραμμα*
- *Να ορίζεις τι είναι η μεταγωγή περιβάλλοντος*
- *Να εξηγείς τι είναι η ελαφρή διεργασία και σε τι διαφέρει από μία διεργασία*
- *Να προσδιορίζεις πότε είναι προτιμότερο να χρησιμοποιούμε τις ελαφρές διεργασίες*

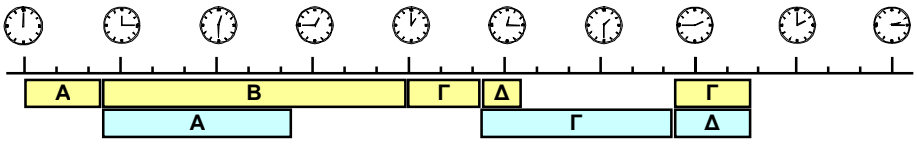
Στους περισσότερους υπολογιστές που έχουν μια μόνο ΚΜΕ, σε κάθε χρονική στιγμή μπορεί να εκτελείται μια μόνο εντολή γλώσσας μηχανής, άρα και ένα μόνο πρόγραμμα. Είναι όμως δυνατόν να κατανέμει η ΚΜΕ το χρόνο της, δίνοντας εκ περιτροπής ένα μικρό ποσοστό του χρόνου της σε κάθε ένα πρόγραμμα έτσι ώστε να δίνει την εντύπωση στους χρήστες ότι τα εκτελεί όλα μαζί.

Ας υποθέσουμε ότι ο ζαχαροπλάστης (η ΚΜΕ) έχει να παρασκευάσει τέσσερα γλυκά, και οι οδηγίες τους αναφέρουν ότι απαιτούν: το Α 35' (10' προετοιμασία και 25' ψήσιμο), το Β 40', το Γ 45' (10' προετοιμασία, 25' ψήσιμο και 10' ολοκλήρωση) και το Δ 15' (5' προετοιμασία και 10' ψήσιμο) αντίστοιχα. Αν η εκτέλεση των συνταγών γίνει με τη σειρά παραλαβής τους, χωρίς παράλληλη χρησιμοποίηση του φούρνου, το όλο έργο μπορεί να οργανωθεί στις φάσεις του Διαγράμματος 1.



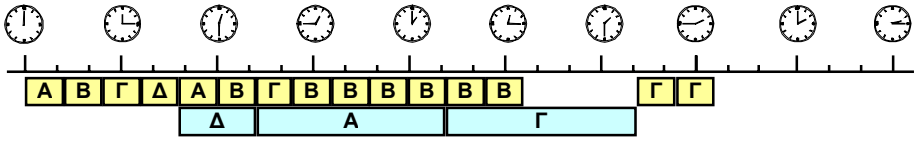
Διάγραμμα 1

Αν η εκτέλεση των συνταγών γίνεται με τη σειρά παραλαβής, αλλά με παράλληλη προετοιμασία και χρήση του φούρνου, έχουμε σημαντική βελτίωση στον απαιτούμενο χρόνο, όπως βλέπουμε στο Διάγραμμα 2.



Διάγραμμα 2

Αν όμως ο ζαχαροπλάστης αφιερώνει 5’ σε κάθε γλυκό, τότε το πρόγραμμα παρασκευής τους μπορεί να οργανωθεί σαν το Διάγραμμα 3.



Διάγραμμα 3

Ο τρόπος αυτός παρασκευής των γλυκών έχει ως αποτέλεσμα τη γρηγορότερη ολοκλήρωση αυτών που παρασκευάζονται εύκολα, ανεξάρτητα από τη σειρά με την οποία δόθηκαν οι συνταγές.

Αντίστοιχα ένα λειτουργικό σύστημα καταμερισμού χρόνου θα μοιράζει το χρόνο της ΚΜΕ σε μικρά στοιχειώδη διαστήματα, τα *κβάντα χρόνου* (time quanta). Σε κάθε κβάντο επιλέγεται ένα από τα προγράμματα και εκτελούνται εντολές από αυτό μέχρι να τελειώσει το χρονικό διάστημα που του διατέθηκε. Όταν ο χρόνος ολοκληρωθεί, επιλέγεται ένα άλλο πρόγραμμα για εκτέλεση κ.ο.κ.

Η διαδικασία αυτή είναι αρκετά πολύπλοκη και επιβαρύνει τις απαιτήσεις του ίδιου του ΛΣ. Το όφελος όμως από αυτή είναι (1) **η αύξηση της απόδοσης**, αφού το υλικό χρησιμοποιείται παράλληλα και (2) **η μείωση του χρόνου ανακύκλωσης**, αφού τα προγράμματα κατά μέσο όρο εξυπηρετούνται πιο γρήγορα.

Η συνεχής εναλλαγή ανάμεσα σε πολλές συνταγές είναι πολύ εύκολο να προκαλέσει σφάλματα. Για να μη γίνονται λοιπόν λάθη, πρέπει ο ζαχαροπλάστης κατά την εναλλαγή από τη μια συνταγή στην άλλη να κάνει ορισμένες ενέργειες όπως:

– Να σημειώνει μέχρι ποιο σημείο της τρέχουσας συνταγής έχει φθάσει, ώστε να μπορεί να συνεχίσει από το σωστό σημείο όταν έλθει ξανά η σειρά της.

– Να τοποθετήσει ό,τι από το γλυκό έχει παρασκευάσει μέχρι εκείνη τη στιγμή σε ένα ορισμένο μέρος, για να μπορεί αργότερα να το ξαναπάρει.

– Αν κάποιο βήμα της τρέχουσας συνταγής δεν είναι δυνατόν να διακοπεί, όπως π.χ. όταν κάτι ψήνεται στο φούρνο, πρέπει να σημειώσει ότι αυτό το βήμα εκκρεμεί. Όταν το ψήσιμο ολοκληρωθεί, θα τοποθετήσει το ψημένο γλυκό μαζί με τα υπόλοιπα υλικά που θα ολοκληρώσουν την παρασκευή του γλυκού και θα βάλει στο φούρνο ό,τι από την επόμενη συνταγή μέχρι τώρα βρισκόταν σε αναμονή.

Όπως στο παράδειγμα του ζαχαροπλαστείου, έτσι και στα υπολογιστικά συστήματα πρέπει να υπάρχει ένα σύνολο πληροφοριών που περιγράφουν την κατάσταση στην οποία βρίσκεται το πρόγραμμα που εκτελείται, για να μπορεί να συνεχιστεί η εκτέλεσή του από το σωστό σημείο την επόμενη φορά που θα έρθει η σειρά του.

Βλέπουμε λοιπόν ότι ένα ΛΣ διαχειρίζεται προγράμματα σε εκτέλεση· μπορεί π.χ. να εκτελεί πολλές φορές το ίδιο πρόγραμμα, μία φορά για κάθε χρήστη. Τα εκτελούμενα προγράμματα αποτελούν ανεξάρτητες οντότητες για το ΛΣ και ονομάζονται *διεργασίες* (processes).

Μια διεργασία είναι ένα πρόγραμμα ή ένα αυτόνομο τμήμα προγράμματος υπό εκτέλεση. Οι όροι *πρόγραμμα* και *διεργασία* διαφοροποιούνται από το γεγονός ότι το πρόγραμμα είναι παθητική οντότητα ενώ η διεργασία είναι ενεργητική.

Η εναλλαγή από τη μια διεργασία στην άλλη ονομάζεται *μεταγωγή περιβάλλοντος* (context switching). Οι πληροφορίες που κρατούνται κατά τη μεταγωγή περιβάλλοντος αφορούν τα ακόλουθα:

- Από ποια εντολή του προγράμματος πρέπει να συνεχισθεί η εκτέλεση της διεργασίας την επόμενη φορά
- Ποια είναι η κατάσταση της ΚΜΕ, ώστε να επαναφερθεί για να συνεχιστεί σωστά η εκτέλεση της διεργασίας
- Ποια είναι τα ενδιάμεσα αποτελέσματα που έχουν υπολογιστεί μέχρι εκείνη τη στιγμή

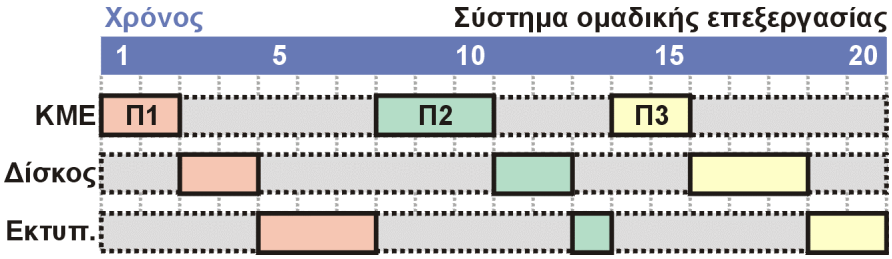
Το ΛΣ καταγράφει αυτές (και άλλες) πληροφορίες για κάθε διεργασία σε μια ειδική περιοχή της μνήμης που ονομάζεται *Σύνολο Ελέγχου Διεργασίας* (Process Control Block), ή αλλιώς *ΣΕΔ* (PCB).

Σύγκριση επίδοσης λειτουργικών συστημάτων

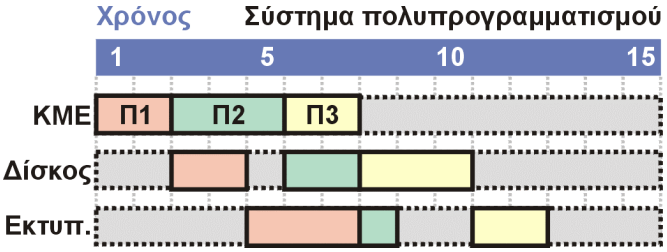
Ένα απλοποιημένο παράδειγμα μπορεί να δείξει με παραστατικό τρόπο τις διαφορές μεταξύ των διαφόρων ΛΣ και τα πλεονεκτήματα του καταμερισμού χρόνου. Σε ένα υπολογιστικό σύστημα δίνονται για εκτέλεση τρία προγράμματα με απαιτήσεις σε χρόνο που φαίνονται στον παράπλευρο πίνακα. Κάνουμε την υπόθεση ότι κάθε διεργασία χρησιμοποιεί κατά σειρά πρώτα την ΚΜΕ, μετά το δίσκο και μετά τον εκτυπωτή.

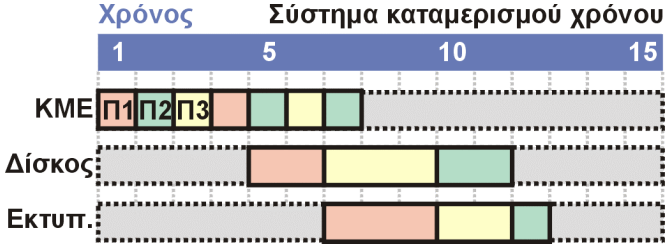
	ΚΜΕ	Δίσκος	Εκτυπωτής	Σύνολο
Πρόγ.1	2	2	3	7
Πρόγ.2	3	2	1	6
Πρόγ.3	2	3	2	7
Σύνολο	7	7	6	20

Το σχεδιάγραμμα εκτέλεσης για ένα ΛΣ ομαδικής επεξεργασίας είναι το εξής:



Για το σύστημα πολυπρογραμματισμού και το σύστημα καταμερισμού χρόνου θα έχουμε:





	Συνολικός χρόνος	Ανενεργός χρόνος	Τέλος της π_1	Τέλος της π_2	Τέλος της π_3
Ομαδική Επεξεργασία	20	40	7	13	20
Πολυπρογραμματισμός	12	16	7	8	12
Καταμερισμός χρόνου	12	16	9	12	11

Τα αποτελέσματα για τα τρια συστήματα συνοψίζονται στον παράπλευρο πίνακα. Παρατηρούμε ότι ο συνολικός χρόνος εκτέλεσης και ο ανενεργός χρόνος για το σύστημα ομαδικής επεξεργασίας είναι πολύ μεγαλύτεροι από ό,τι στα άλλα δυο συστήματα. Το σύστημα καταμερισμού χρόνου έχει άλλο ένα πλεονέκτημα: καμία διεργασία δεν ευνοείται ως προς το πότε θα τερματίσει (ενώ στο σύστημα πολυπρογραμματισμού υπάρχει τέτοια ανισότητα)· μια και όλες οι διεργασίες έχουν περίπου τις ίδιες χρονικές απαιτήσεις, τελειώνουν περίπου στον ίδιο χρόνο.

Οι επιδόσεις της εκτέλεσης των διαφόρων διεργασιών από ένα ΛΣ εξαρτάται από τον τύπο του ΛΣ, αλλά επηρεάζεται και από τις απαιτήσεις των διεργασιών.

Ελαφρές διεργασίες

Κάθε ΛΣ αποτελείται από πολλά επί μέρους προγράμματα, τα οποία εκτελούνται παράλληλα για να εξυπηρετούν τους χρήστες, είναι δηλαδή και αυτό χωρισμένο σε διεργασίες. Το ίδιο μπορεί να γίνει και με τα προγράμματα των χρηστών, να διαιρεθούν δηλαδή σε τμήματα τα οποία εκτελούνται παράλληλα.

Ο ζαχαροπλάστης (ΚΜΕ) αφιερώνει 5’ κάθε φορά στην εκτέλεση μιας συνταγής. Αυτά τα 5’ μπορεί να τα μοιράσει σε 5 τμήματα του 1’ το καθένα. Σε κάθε τέτοιο τμήμα του 1’ μπορεί να ασχοληθεί σε κάθε τμήμα με ένα διαφορετικό στοιχείο του γλυκού το οποίο είναι ανεξάρτητο από τα άλλα. Έτσι μπορεί π.χ. να παρακολουθεί το ψήσιμο της ζύμης στο φούρνο και να ανακατεύει την κρέμα που βράζει.

Όταν όμως τα τμήματα αυτά πρέπει να μοιράζονται διάφορα στοιχεία του προγράμματος, όπως π.χ. μεταβλητές, μπορεί να χρησιμοποιηθεί και ένα εναλλακτικό είδος διεργασιών, οι *ελαφρές διεργασίες* (lightweight processes) ή *νήματα εκτέλεσης* (threads of execution) ή απλούστερα *νήματα* (threads).

Τα νήματα έχουν πολλές ομοιότητες με τις διεργασίες και μια βασική διαφορά: χρησιμοποιούν από κοινού ένα τμήμα μνήμης, στο οποίο έχουν όλα πρόσβαση.

Όπως για κάθε διεργασία, έτσι και για κάθε νήμα το ΛΣ κρατά ένα σύνολο από πληροφορίες σε μια ειδική περιοχή της μνήμης, η οποία αποκαλείται *Σύνολο Ελέγχου Νήματος* (Thread Control Block) ή αλλιώς *ΣΕΝ* (TCB). Το ΣΕΝ κρατά τις απαραίτητες πληροφορίες για κάθε νήμα, αλλά όσες πληροφορίες είναι κοινές μεταξύ τους (π.χ. αυτές για τη μοιραζόμενη μνήμη) καταγράφονται μια φορά μόνο.

Ο ζαχαροπλάστης πρέπει να θυμάται ή να καταγράφει την πρόοδο για κάθε στάδιο μιας συνταγής που εκτελείται παράλληλα με άλλα στάδια της εκτέλεσής της. Τα σκεύη όμως και τα υλικά που αφορούν τα διάφορα στάδια τα τοποθετεί στο ίδιο μέρος, εκείνο που έχει καθορίσει για τη συγκεκριμένη συνταγή.

Όταν ένα πρόγραμμα διαιρεθεί σε νήματα που εκτελούνται «παράλληλα» (όπως οι διεργασίες), αντί σε ανεξάρτητες διεργασίες, προκαλείται μικρότερη επιβάρυνση στο υπολογιστικό σύστημα, γιατί η μεταγωγή περιβάλλοντος μεταξύ νημάτων είναι πιο γρήγορη από ό,τι μεταξύ διεργασιών. Επιπλέον δεν είναι απαραίτητο πάντα να χρησιμοποιηθούν μηχανισμοί επικοινωνίας από τα νήματα, αφού έχουν για το σκοπό αυτό τη μνήμη που μοιράζονται, κάτι που θα γίνει κατανοητό παρακάτω.

Ένα παράδειγμα προγράμματος που μπορεί να χωριστεί σε νήματα, φαίνεται στο σχήμα. Το πρόγραμμα αυτό υπολογίζει για τις μεταβλητές x και y:

- Με τη ρουτίνα Athroisma, το άθροισμα τους και το αποθηκεύει στη μεταβλητή sum.
- Με τη ρουτίνα Diafora, τη διαφορά τους και την αποθηκεύει στη μεταβλητή diff.
- Με τη ρουτίνα Phliko, το πηλίκο τους και το αποθηκεύει στη μεταβλητή quot.

```
var
x, y: integer;
sum, diff: integer;
quot: real;

procedure Athroisma;
begin
   sum := x + y
end;

procedure Diafora;
begin
   diff := x - y
end;

procedure Phliko;
begin
   quot := x / y
end;
```

Οι τρεις αυτοί υπολογισμοί προσδιορίζουν την τιμή διαφορετικών μεταβλητών και ο ένας δεν εξαρτάται από τον άλλο, έτσι μπορούν να γίνουν παράλληλα.

Αν επιλέξουμε να διαιρέσουμε το πρόγραμμα σε τρεις ταυτόχρονες διεργασίες, και οι τρεις χρειάζονται τις μεταβλητές x και y για να κάνουν τους υπολογισμούς τους. Έτσι η λύση αυτή έχει δυο μειονεκτήματα:

→ Η μνήμη σπαταλάται, γιατί έχουμε τρία διαφορετικά αντίγραφα κάθε μεταβλητής στη μνήμη. Εδώ βέβαια πρόκειται για δυο μεταβλητές μόνο, αλλά αν επρόκειτο για πολλά δεδομένα, η σπατάλη θα ήταν πολύ ουσιαστική.

→ Πρέπει να χρησιμοποιηθεί κάποιος μηχανισμός επικοινωνίας που θα εξασφαλίζει ότι οι μεταβλητές x και y των τριών διεργασιών περιέχουν τα ίδια δεδομένα. Αν π.χ. τα δεδομένα δίνονται από το χρήστη, πρέπει η μια από τις τρεις διεργασίες να τα διαβάσει και να τα στείλει στις άλλες δυο, ή μια άλλη διεργασία να τα ζητήσει από το χρήστη και να τα γράψει σε ένα αρχείο, από όπου θα τα διαβάσουν οι υπόλοιπες. Αυτές οι διαδικασίες και προγραμματιστικό κόπο απαιτούν, και επιβαρύνουν το χρόνο εκτέλεσης των διεργασιών, και απαιτούν επιπλέον πόρους του συστήματος (μηχανισμούς επικοινωνίας ή αρχεία).

Η εναλλακτική λύση που δεν έχει τα παραπάνω μειονεκτήματα είναι να χωρίσουμε το πρόγραμμα σε τρία *νήματα*. Τα νήματα αυτά θα μοιράζονται τη μνήμη που καταλαμβάνουν οι δυο μεταβλητές, και θα τη διαβάζουν για να κάνουν τους υπολογισμούς τους. Μόνο ένα αντίγραφο των μεταβλητών θα κρατείται στη μνήμη -έτσι δε γίνεται σπατάλη- και βέβαια δεν τίθεται ζήτημα επικοινωνίας για ανταλλαγή δεδομένων.

Τα προγράμματα που μπορούν να διαιρεθούν σε τμήματα, κάθε ένα από τα οποία ανατίθεται σε ένα νήμα, ονομάζονται *ταυτόχρονα* (concurrent programs). Ένα τέτοιο πρόγραμμα είναι το προηγούμενο, ή π.χ. το σύστημα κράτησης θέσεων μιας αεροπορικής εταιρίας, όπου κάθε πράκτορας αντιστοιχεί σε ένα νήμα.

Στα επόμενα μαθήματα, ο όρος «Διεργασία» θα χρησιμοποιείται τόσο για τις διεργασίες (που δε χρησιμοποιούν μνήμη από κοινού) όσο και για τα νήματα (τα οποία χρησιμοποιούν μνήμη από κοινού), εκτός και αν αναφέρεται διαφορετικά.

Ανακεφαλαίωση

Ο καταμερισμός του χρόνου της ΚΜΕ μεταξύ πολλών προγραμμάτων δίνει τη δυνατότητα να εκτελούνται αυτά ταυτόχρονα βελτιώνοντας συνολικά τις επιδόσεις του υπολογιστικού συστήματος. Τα προγράμματα που βρίσκονται σε κάποιο στάδιο της εκτέλεσής τους από την ΚΜΕ ονομάζονται διεργασίες. Η διαδικασία εναλλαγής από τη μια διεργασία στην άλλη, δηλαδή η μεταγωγή περιβάλλοντος, απαιτεί την καταγραφή πληροφοριών για τη διεργασία που διακόπτεται ώστε να μπορεί να συνεχιστεί αργότερα σωστά η εκτέλεσή της.

Μια εναλλακτική μορφή οργάνωσης ταυτόχρονων προγραμμάτων, τα οποία διαιρούνται σε τμήματα που μπορούν να εκτελούνται παράλληλα, είναι τα νήματα, μια «ελαφριά» μορφή διεργασιών. Τα νήματα χρησιμοποιούν κοινό τμήμα μνήμης και έτσι δεν απαιτούν μηχανισμούς επικοινωνίας πολλές φορές· επίσης επιβαρύνουν λιγότερο το υπολογιστικό σύστημα.

Γλωσσάριο όρων

Διεργασία	*Process*
Ελαφρή διεργασία	*Lightweight Process*
Κβάντο Χρόνου	*Time Quantum πληθ.-α*
Μεταγωγή Περιβάλλοντος	*Context Switching*
Νήμα	*Thread*
Νήμα Εκτέλεσης	*Thread of Execution*
Σύνολο Ελέγχου Διεργασίας - ΣΕΔ	*Process Control Block - PCB*
Σύνολο Ελέγχου Νήματος - ΣΕΝ	*Thread Control Block - TCB*
Ταυτόχρονο Πρόγραμμα	*Concurrent Program*

Ερωτήσεις

? Ποια είναι τα οφέλη και οι συνέπειες όταν το ΛΣ εναλλάσσει προγράμματα στη χρήση της ΚΜΕ;

? Σε τι διαφέρει η διεργασία από ένα πρόγραμμα; Μπορούν πολλές διεργασίες να αντιστοιχούν στο ίδιο πρόγραμμα;

? Ποια είναι η βασική διαφορά μίας διεργασίας με μία ελαφρή διεργασία;

? Τι είναι η μεταγωγή περιβάλλοντος για μία διεργασία; Είναι η ίδια με τη μεταγωγή περιβάλλοντος για μία ελαφρή διεργασία;

? Πότε συμφέρει να χρησιμοποιήσουμε τις ελαφρές διεργασίες; Πότε νομίζεις ότι πρέπει να τις αποφύγουμε;

Μάθημα 7.3 Απεικόνιση Διεργασιών

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει δύο μεθόδους απεικόνισης ταυτόχρονων διεργασιών, μία σχηματική και μία προγραμματιστική.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τι είναι ο γράφος προβαδίσματος*
- *Να αποφασίζεις πότε ένας γράφος προβαδίσματος παριστάνει διεργασίες που δεν μπορούν να εκτελεστούν*
- *Να συμβολίζεις παράλληλες διεργασίες με τις εντολές parbegin και parend*
- *Να περιγράφεις πώς φτιάχνουμε ένα πρόγραμμα με τις εντολές parbegin και parend από ένα γράφο προβαδίσματος*

Ένας τρόπος για να αναπαραστήσουμε γραφικά τα ταυτόχρονα προγράμματα είναι να χρησιμοποιήσουμε το *γράφο προβαδίσματος* (precedence graph). Με τη βοήθεια ενός τέτοιου γράφου που αναπαριστά κάποιο πρόγραμμα, μπορούμε να δούμε ποια τμήματα του προγράμματος μπορούν να εκτελεστούν παράλληλα και ποια πρέπει να περιμένουν μέχρι να ολοκληρωθούν κάποια άλλα.

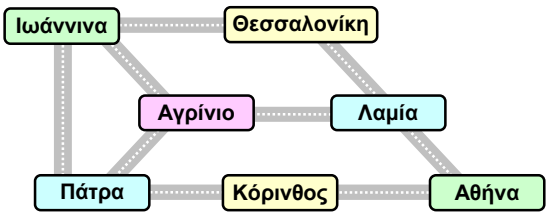
> Η παρασκευή ενός γλυκού αποτελείται από τα εξής στάδια: χτύπημα μαρέγκας, ανακάτεμα των υπόλοιπων υλικών, κόψιμο ξηρών καρπών, ανάμιξη μαρέγκας με τα υπόλοιπα υλικά, ανάμιξη ξηρών καρπών στη ζύμη, ψήσιμο του γλυκού, προετοιμασία γλάσου*, γλασάρισμα του γλυκού. Κάθε ένα από τα στάδια αυτά μπορεί να αντιστοιχιστεί σε μια διαφορετική διεργασία. Για να οργανώσει όσο πιο αποδοτικά γίνεται ο ζαχαροπλάστης την εργασία του, πρέπει να γνωρίζει ποια στάδια πρέπει να έχουν ολοκληρωθεί προτού να ξεκινήσει ένα άλλο. Για παράδειγμα, δεν μπορεί να γίνει το ψήσιμο του γλυκού αν δεν έχει ετοιμαστεί πρώτα η ζύμη.

Ένας γράφος αποτελείται από *κόμβους* (vertices) και ακμές (edges). Μια ακμή συνδέει δυο κόμβους μεταξύ τους, και μπορεί να έχει ή όχι κατεύθυνση. Στην περίπτωση που η ακμή έχει κατεύθυνση ονομάζεται *κατευθυνόμενη* (directed).

Συνήθως σε ένα γράφο δεν εμφανίζονται μαζί κατευθυνόμενες και μη ακμές· όταν όλες οι ακμές είναι κατευθυνόμενες πρόκειται για ένα *κατευθυνόμενο γράφο* (directed graph), αλλιώς αποκαλείται απλώς γράφος.

---

* Γλάσο: μίγμα συνήθως από βούτυρο, ζάχαρη και αρωματικές ύλες που χρησιμοποιείται για την επικάλυψη των γλυκών και έχει γυαλιστερή όψη.



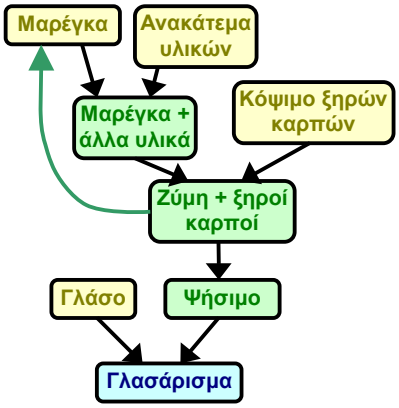
Ένας καλός τρόπος για να φανταστούμε ένα γράφο είναι το οδικό δίκτυο μεταξύ πόλεων. Οι κόμβοι είναι οι πόλεις και οι ακμές είναι οι δρόμοι που συνδέουν ζεύγη πόλεων. Οι μονόδρομοι είναι κατευθυνόμενες ακμές, και οι δρόμοι διπλής κατευθύνσεως είναι μη κατευθυνόμενες ακμές.

Στο σχήμα οι κόμβοι του γράφου είναι οι «Αγρίνιο», «Αθήνα», «Θεσσαλονίκη», «Ιωάννινα», «Κόρινθος», «Λαμία» και «Πάτρα». Όλοι οι δρόμοι του γράφου είναι διπλής κατευθύνσεως, οπότε ο γράφος είναι απλός και κάθε ακμή μπορούμε να τη «διασχίσουμε» και προς τις δυο κατευθύνσεις. Αν είχε κάποιους μονοδρόμους, τότε θα αντικαθιστούσαμε όλες τις απλές ακμές με δυο αντίθετης κατευθύνσεως. Έτσι για το δρόμο Θεσσαλονίκης-Ιωαννίνων, θα είχαμε μια ακμή Θεσσαλονίκη → Ιωάννινα και μια Ιωάννινα → Θεσσαλονίκη.

Σε ένα γράφο προβαδίσματος, κάθε διεργασία παριστάνεται με ένα κόμβο. Αν η διεργασία $Δ_x$ πρέπει να έχει ολοκληρωθεί για να αρχίσει η $Δ_y$, τότε ο γράφος περιέχει μια κατευθυνόμενη ακμή από τον κόμβο της $Δ_x$ σε αυτόν της $Δ_y$. Στην εποπτική αναπαράσταση του γράφου έχουμε ένα βέλος από τη $Δ_x$ προς τη $Δ_y$.

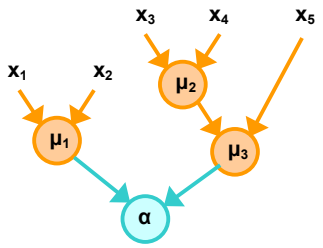
Οι γράφοι προβαδίσματος είναι λοιπόν κατευθυνόμενοι και μια βασική ιδιότητά τους είναι ότι δεν έχουν κύκλους, είναι δηλαδή *ακυκλικοί* (acyclic). Ένας κύκλος είναι μια σειρά από ακμές την οποία μπορούμε να διασχίσουμε από κόμβο σε κόμβο, καταλήγοντας στο σημείο από όπου ξεκινήσαμε. Στο σχήμα του οδικού δικτύου ένας τέτοιος κύκλος είναι ο Κόρινθος → Πάτρα → Αγρίνιο → Λαμία → Αθήνα → Κόρινθος.

Αν ένας γράφος προβαδίσματος έχει κύκλους, τότε δεν μπορεί να βρεθεί καμία δυνατή σειρά εκτέλεσης των διεργασιών.



Ο γράφος προβαδίσματος για το γλυκό που περιγράψαμε δίνεται στο σχήμα. Αν στο γράφο αυτό υπάρχει και μια ακμή ακόμα (η πράσινη γραμμή) από τη διεργασία «Ζύμη+ξηροί καρποί» στη διεργασία «Μαρέγκα», η οποία δημιουργεί ένα κύκλο θα έπρεπε: Η ανάμιξη «Ζύμη+ξηροί καρποί» να γίνει πριν από τη «Μαρέγκα», η οποία πρέπει να γίνει πριν από την ανάμιξη «Μαρέγκα+άλλα υλικά», που πρέπει να γίνει πριν από τη «Ζύμη+ξηροί καρποί». Είναι φανερό ότι δεν είναι δυνατόν να παρασκευαστεί το γλυκό στην περίπτωση αυτή, εξαιτίας του κύκλου στο γράφο προβαδίσματος.

Σε ένα γράφο προβαδίσματος είναι εύκολο να διαπιστώσουμε αν δυο διεργασίες μπορούν να εκτελεστούν ταυτόχρονα, αν δεν είναι δυνατό να μεταβούμε από τον κόμβο της μιας στον κόμβο της άλλης. Φυσικά η κυκλοφορία επιτρέπεται μόνο πάνω στις ακμές και με τη φορά που δείχνουν τα βέλη.



Ένα παράδειγμα ταυτόχρονου προγράμματος σε ένα υπολογιστή είναι η αριθμητική έκφραση $x_1 \times x_2 - (x_3 - x_4) / x_5$. Αν αναλύσουμε την έκφραση σε μερικά αποτελέσματα έχουμε $μ_1 := x_1 \times x_2$, $μ_2 := x_3 - x_4$, $μ_3 := μ_2 / x_5$, $α := μ_1 - μ_3$. Για να υπολογιστεί ένα μερικό αποτέλεσμα, πρέπει να είναι διαθέσιμες όλες οι τιμές που χρησιμοποιεί, π.χ. για να υπολογιστεί το $μ_3$ χρειάζονται οι τιμές των $μ_2$ και x_5.

Αν αντιστοιχίσουμε κάθε μερικό αποτέλεσμα σε μια διεργασία, η εξάρτηση των μερικών αποτελεσμάτων από άλλα μας δίνει το γράφο προβαδίσματος για την έκφραση, από όπου φαίνεται ότι π.χ. τα $μ_1$ και $μ_2$ ή τα $μ_1$ και $μ_3$ μπορούν να υπολογιστούν ταυτόχρονα, γιατί δε συνδέονται με κάποιο μονοπάτι, ενώ τα $μ_2$ και $μ_3$ που συνδέονται δεν μπορούν να υπολογιστούν ταυτόχρονα.

Ο γράφος προβαδίσματος είναι ένα πολύ καλό εργαλείο για τη γραφική αναπαράσταση ταυτόχρονων προγραμμάτων. Οι συνηθισμένες όμως γλώσσες προγραμματισμού δεν είναι φτιαγμένες έτσι ώστε να μπορούν να περιγράψουν ταυτόχρονα προγράμματα και επομένως ούτε και γράφους προβαδίσματος. Για να συμβεί αυτό, θα πρέπει οι γλώσσες προγραμματισμού να εμπλουτιστούν κατάλληλα με νέα στοιχεία, όπως θα δούμε στη συνέχεια.

Ο συμβολισμός parbegin … parend

Οι εντολές *parbegin* (**par**allel **begin**) και *parend* (**par**allel **end**) είναι εντολές υψηλού επιπέδου για την περιγραφή ταυτοχρονισμού που προτάθηκαν από τον Dijkstra το 1965.

Σε ένα πρόγραμμα π.χ. Pascal τα **begin** … **end** περικλείουν μια ομάδα εντολών που πρέπει να εκτελεστεί σειριακά. Οι εντολές της ομάδας χωρίζονται μεταξύ τους με ερωτηματικό («;»). Αντίστοιχα, μια ομάδα εντολών που μπορεί να εκτελεστεί παράλληλα περικλείεται από τα **parbegin** … **parend** και οι εντολές χωρίζονται μεταξύ τους με το σύμβολο «||». Οι εντολές αυτές μπορούν να είναι απλές εντολές ή σύνθετες, να είναι δηλαδή με τη σειρά τους μια ομάδα εντολών που περικλείονται από begin…end.

```
begin
   m1 := x1 * x2;
   m2 := x3 - x4;
   m3 := m2 / x5;
   r := m1 - m3
end
```

Σχήμα Α

Η εκτέλεση μιας ομάδας παραλλήλων εντολών που περικλείονται από τα parbegin … parend γίνεται ταυτόχρονα: ξεκινούν όλες μαζί και η ομάδα εντολών τελειώνει όταν όλες τους τερματίσουν.

Η παράλληλη υλοποίηση του προγράμματος του Σχήματος Α φαίνεται στο Σχήμα Β.

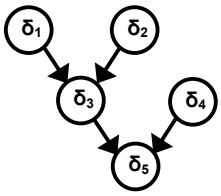
```
begin
   parbegin
      m1 := x1 * x2 ||
      begin
         m2 := x3 - x4;
         m3 := m2 / x5
      end;
   parend;
   r := m1 - m3
end
```

Σχήμα Β

Οι εντολές parbegin … parend δεν είναι σε θέση να περιγράψουν όλους τους δυνατούς γράφους προβαδίσματος· έχουν όμως το πλεονέκτημα ότι κάνουν το παράλληλο πρόγραμμα πολύ ευανάγνωστο και κατανοητό.

Γράφοι προβαδίσματος και προγράμματα

Πολλοί γράφοι προβαδίσματος μπορούν να αντιστοιχιστούν με περισσότερα από ένα προγράμματα γραμμένα με τις εντολές parbegin-parend. Στο γράφο του σχήματος, η εντολή δ_4 μπορεί να εκτελεστεί ταυτόχρονα με οποιαδήποτε από τις δ_1, δ_2 και δ_3, γιατί δεν υπάρχει μονοπάτι μέσα στο γράφο που να τις συνδέει.Έτσι μπορούν να γραφούν τουλάχιστον *τρία* διαφορετικά παράλληλα προγράμματα τα οποία υπακούουν στους περιορισμούς του γράφου.



Οι τρεις διαφορετικές λύσεις είναι:

1) Η δ_4 εκτελείται παράλληλα με τις δ_1 και δ_2 (οι οποίες μπορούν να εκτελεστούν παράλληλα), πριν από την δ_3. Το πρόγραμμα είναι:

```
parbegin δ1 || δ2 || δ4 parend;
δ3; δ5
```

2) Πρώτα εκτελούνται παράλληλα οι δ_1 και δ_2, και στη συνέχεια η δ_3 εκτελείται παράλληλα με την δ_4.

```
parbegin δ1 || δ2 parend;
parbegin δ3 || δ4 parend;
δ5
```

3) Η δ_1 και η δ_2 εκτελούνται παράλληλα, και μόλις τελειώσουν και οι δυο εκτελείται η δ_3. Η δ_4 εκτελείται παράλληλα με ολόκληρη την ομάδα των δ_1, δ_2, δ_3· ξεκινά μαζί με τις δ_1, δ_2, αλλά η δ_3 ξεκινά ανεξάρτητα από αυτήν. Η δ_5 περιμένει τις δ_3 και δ_4 να τερματίσουν για να ξεκινήσει.

```
parbegin
   begin
      parbegin δ1 || δ2 parend;
      δ3;
   end
||
   δ4
parend;
δ5
```

Από τα τρία προγράμματα μόνο το ένα είναι το «σωστό», δηλαδή αυτό που αντιπροσωπεύει καλύτερα το γράφο προβαδίσματος. Το πρώτο πρόγραμμα έχει ένα πρόβλημα: η διεργασία δ_3 περιμένει και την δ_4 να τελειώσει (εκτός από τις δ_1 και δ_2), ενώ αυτό δεν είναι απαραίτητο. Παρομοίως, στο δεύτερο πρόγραμμα η δ_4 περιμένει τις δ_1 και δ_2 χωρίς να υπάρχει λόγος. Και τα δυο προγράμματα δηλαδή, θέτουν επιπλέον περιορισμούς στην εκτέλεση των διεργασιών από αυτούς που περιγράφει ο γράφος. Αυτό δε συμβαίνει στο τρίτο πρόγραμμα, το οποίο είναι και το σωστό για την προγραμματιστική αναπαράσταση του γράφου προβαδίσματος.

Ανακεφαλαίωση

Ένας τρόπος αναπαράστασης ταυτοχρόνων προγραμμάτων είναι ο γράφος προβαδίσματος, όπου κάθε διεργασία είναι ένας κόμβος. Αν μια διεργασία πρέπει να ολοκληρωθεί προτού μια άλλη ξεκινήσει, ο γράφος περιέχει μια κατευθυνόμενη ακμή από τον πρώτο κόμβο στο δεύτερο. Ο γράφος προβαδίσματος πρέπει να είναι ακυκλικός. Διεργασίες που δε συνδέονται με ένα μονοπάτι μέσα στο γράφο προβαδίσματος μπορούν να εκτελεστούν ταυτόχρονα.

Οι εντολές parbegin και parend είναι ένας πολύ απλός και εύχρηστος συμβολισμός που χρησιμοποιείται για την αναπαράσταση ταυτόχρονων διεργασιών

Γλωσσάριο όρων

Ακμή	*Edge*
Ακυκλικός Γράφος	*Acyclic Graph*
Γράφος Προβαδίσματος	*Precedence Graph*
Κατευθυνόμενη Ακμή	*Directed Edge*
Κατευθυνόμενος Γράφος	*Directed Graph*
Κόμβος	*Vertex πληθ. Vertices*
Μονοπάτι	*Path*

Ερωτήσεις

? Τι είναι ο γράφος;

? Τι συμβολίζει μία ακμή σε ένα γράφο προβαδίσματος;

? Τι συμβαίνει αν ένας γράφος προβαδίσματος περιέχει ένα κύκλο από ακμές;

? Πώς εκτελούνται οι εντολές που περικλείονται από ένα ζεύγος parbegin … parend;

? Η αντιστοιχία προγραμμάτων και γράφων προβαδίσματος είναι 1-1; Γιατί;

Μάθημα 7.4 Κρίσιμα Τμήματα και Αμοιβαίος Αποκλεισμός

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να εξηγήσει την έννοια του κρίσιμου τμήματος σε μία διεργασία και να δείξει τη λύση για ένα απλό πρόβλημα κρίσιμου τμήματος.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τι είναι κρίσιμο τμήμα μίας διεργασίας*
- *Να ορίζεις τον αμοιβαίο αποκλεισμό*
- *Να περιγράφεις πώς επιτυγχάνουμε αμοιβαίο αποκλεισμό σε ένα απλό πρόβλημα*

Κρίσιμα τμήματα

Αρκετά από τα εργαλεία που χρησιμοποιεί ο ζαχαροπλάστης, όπως π.χ. ο θερμοθάλαμος, μπορούν να χρησιμοποιηθούν για διαφόρες συνταγές. Κατά τη διάρκεια του πενταλέπτου εκτέλεσης μιας συνταγής, ο ζαχαροπλάστης έχει τοποθετήσει στο θερμοθάλαμο λιωμένη σοκολάτα για να τη διατηρήσει ζεστή, αλλά το πεντάλεπτο της συνταγής εκπνέει. Η συνταγή με την οποία ασχολείται για τα επόμενα 5' ζητά να τοποθετηθεί η ζύμη στο θερμοθάλαμο για να φουσκώσει. Αν ο ζαχαροπλάστης βγάλει τη λιωμένη σοκολάτα για να βάλει τη ζύμη, η σοκολάτα θα κρυώσει, καταστρέφοντας έτσι την εκτέλεση της πρώτης συνταγής.

Οι οδηγίες του ιδιοκτήτη προς το ζαχαροπλάστη (ΚΜΕ), δηλαδή το ΛΣ, πρέπει να μεριμνούν ώστε κάτι τέτοιο να αποφευχθεί.

Σε ένα υπολογιστικό σύστημα, υπάρχουν *μοιραζόμενοι πόροι* (shared resources) -στο παράδειγμά μας ο θερμοθάλαμος- που χρησιμοποιούνται από πολλές διεργασίες και για να διαβαστούν αλλά και για να γραφτούν. Ο τρόπος που γίνονται οι προσπελάσεις σε αυτά έχει πολύ μεγάλη σημασία, γιατί όπως είδαμε και στο παράδειγμα, μια αλλαγή στα δεδομένα από τη μια διεργασία επηρεάζει και τις υπόλοιπες.

Μια τέτοια χαρακτηριστική περίπτωση διεργασιών που μοιράζονται δεδομένα είναι ένα online σύστημα τράπεζας. Η διαδικασία ανάληψης ενός ποσού Α από ένα μηχάνημα ΑΤΜ γίνεται σε πέντε φάσεις:

1. Ρώτα τον πελάτη για το χρηματικό ποσό Α που θέλει
2. Διάβασε το υπόλοιπο Υ του λογαριασμού
3. Υπολόγισε το νέο υπόλοιπο Υ' μετά την ανάληψη του ποσού Α, Υ':=Υ-Α
4. Αποθήκευσε το νέο υπόλοιπο
5. Δώσε τα χρήματα και την απόδειξη

Τι θα γίνει αν γίνονται από δυο διαφορετικά μηχανήματα συγχρόνως αναλήψεις για τον ίδιο λογαριασμό; Ας φανταστούμε ότι σε ένα κοινό λογαριασμό το αρχικό υπόλοιπο του λογαριασμού είναι 200.000, ο ένας πελάτης (π.χ. η σύζυγος) ζητά 50.000, ενώ ο άλλος (ο σύζυγος) ζητά 20.000. Επίσης υποθέτουμε ότι η εναλλαγή των διεργασιών γίνεται με τέτοιο ρυθμό ώστε σε κάθε κβάντο χρόνου, το οποίο έχει διάρκεια 1', μια διεργασία εκτελεί μια από τις τέσσερις παραπάνω φάσεις. Το υπόλοιπο του λογαριασμού μετά από τις δυο αναλήψεις θα πρέπει να είναι βέβαια 130.000.

Ώρα	Διεργασία Α	Διεργασία Β	Λογαριασμός
13:01	Ρώτα για το ποσό A=50.000	-	200.000
13:02	-	Ρώτα για το ποσό A=20.000	200.000
13:03	Διάβασε το υπόλοιπο Y=200.000	-	200.000
13:03	-	Διάβασε το υπόλοιπο Y=200.000	200.000
13:04	Y' := Y - A = 200.000-50.000 = *150.000*	-	200.000
13:05	-	Y' := Y - A = 200.000-20.000 = *180.000*	200.000
13:06	Αποθήκευσε το Y'	-	150.000
13:07	-	Αποθήκευσε το Y'	**180.000**

Βλέπουμε λοιπόν ότι το τελικό υπόλοιπο του λογαριασμού δεν είναι σωστό, και μάλιστα εξαρτάται από το ποια διεργασία ξεκίνησε πρώτη: αν είχε ξεκινήσει πρώτη η Β, το υπόλοιπο θα ήταν 150.000. Αν η διεργασία Α ξεκινήσει πρώτη, ο πελάτης που αντιστοιχεί σε αυτή θα λάβει απόδειξη με το σωστό υπόλοιπο για τη συναλλαγή του, ενώ ο πελάτης της διεργασίας Β θα λάβει απόδειξη με λανθασμένο υπόλοιπο.

Το λάθος στο πρόγραμμα προέρχεται από το γεγονός ότι πολλές διεργασίες μοιράζονται δεδομένα. Αν διαβάζουν και γράφουν σε αυτά ανεξέλεγκτα, τότε τα δεδομένα δεν παίρνουν τις σωστές τιμές. Το πρόβλημα αυτό ονομάζεται το πρόβλημα του *κρισίμου τμήματος* (critical section) όπου πρέπει να εξασφαλισθεί η *ακεραιότητα των δεδομένων* (data integrity).

Για να λύσει το πρόβλημα με το θερμοθάλαμο, οι οδηγίες προς το ζαχαροπλάστη (ΛΣ) μπορούν να προβλέπουν δυο λύσεις:

- Παρότι το πεντάλεπτο της πρώτης συνταγής έχει τελειώσει, να συνεχίσει να ασχολείται με αυτή μέχρι να έρθει η στιγμή που χρησιμοποιεί τη σοκολάτα και να απελευθερώσει το θερμοθάλαμο. Τότε θα τη διακόψει και θα προχωρήσει στην επόμενη.
- Να σημειώσει ότι ο θερμοθάλαμος είναι δεσμευμένος από την πρώτη συνταγή. Έτσι αφού η δεύτερη συνταγή χρειάζεται το θερμοθάλαμο για να προχωρήσει, και αυτός δεν είναι ελεύθερος, δεν προχωρεί με αυτή. Τη διακόπτει δηλαδή, πριν ακόμα να εκπνεύσει το πεντάλεπτό της και προχωρεί με κάποια άλλη. Το ίδιο συμβαίνει για όσες συνταγές και όσες φορές χρειάζονται το θερμοθάλαμο, έως ότου αυτός απελευθερωθεί.

Δυο αντίστοιχες λύσεις μπορούν να εφαρμοστούν και στο πρόβλημα του τραπεζικού λογαριασμού:

- Μέχρι να ολοκληρωθεί η συναλλαγή του πρώτου πελάτη δεν ξεκινά αυτή του δεύτερου. *Υλοποίηση*: Αδρανοποιείται το σύστημα διακοπών του χρονιστή (ο οποίος είναι υπεύθυνος να ενημερώνει την ΚΜΕ κάθε φορά που πρέπει να

διακοπεί μια διεργασία για να εκτελεστεί άλλη) μέχρι η πρώτη συναλλαγή να ολοκληρωθεί, οπότε δεν είναι δυνατόν να διακοπεί αυτή για να εκτελεστεί κάποια άλλη.

- Το ΛΣ «σημειώνει» ότι ο λογαριασμός χρησιμοποιείται από την πρώτη διεργασία, και η δεύτερη περιμένει έως ότου ο λογαριασμός «αποσημειωθεί», ελευθερωθεί δηλαδή. *Υλοποίηση*: Με ειδικούς μηχανισμούς, όπως οι σηματοφορείς, που θα εξεταστούν σε επόμενο μάθημα.

Και στις δυο περιπτώσεις, το πρόβλημα της ακεραιότητας των δεδομένων δεν αφορά την πρώτη φάση της διαδικασίας ανάληψης, δηλαδή την εισαγωγή από τον πελάτη του χρηματικού ποσού που επιθυμεί, γιατί τότε τα δεδομένα του λογαριασμού δεν έχουν διαβαστεί ή αλλάξει ακόμα. Αν διακοπεί η διεργασία αμέσως μόλις ρωτήσει τον πελάτη για το ύψος του ποσού, ό,τι αλλαγές γίνουν στο λογαριασμό από κάποια άλλη συναλλαγή δε θα επηρεάσουν αυτήν.

Βλέπουμε λοιπόν ότι μόνο σε ένα τμήμα της κάθε διεργασίας εμφανίζεται κίνδυνος για την ακεραιότητα των δεδομένων. Το τμήμα αυτό του κώδικα ονομάζεται *κρίσιμο τμήμα*.

Όταν μια διεργασία εκτελεί τον κώδικα του κρίσιμου τμήματός της, καμία άλλη δεν μπορεί να εκτελεί το δικό της κρίσιμο τμήμα, έχουμε δηλαδή το φαινόμενο του *αμοιβαίου αποκλεισμού* (mutual exclusion).

Αν δυο ή περισσότερες διεργασίες είναι ταυτόχρονα έτοιμες να εισέλθουν στο κρίσιμο τμήμα τους, η επιλογή αυτής που τελικά θα το εκτελέσει γίνεται με τυχαίο τρόπο.

Ο Dijkstra (1965) έθεσε μερικούς ακόμα περιορισμούς για τα κρίσιμα τμήματα:

1. Όταν μια διεργασία εκτελεί κώδικα εκτός του κρίσιμου τμήματός της, δεν μπορεί να αποτρέψει άλλες διεργασίες να εκτελέσουν το δικό τους κρίσιμο τμήμα.
2. Όταν δυο ή περισσότερες διεργασίες είναι ταυτόχρονα έτοιμες να εισέλθουν στο κρίσιμο τμήμα τους, η λήψη απόφασης δεν πρέπει να αναβάλλεται επ' άπειρο.
3. Πρέπει να υπάρχει *δικαιοσύνη* (fairness) στον τρόπο επιλογής της διεργασίας που θα εισέλθει στο κρίσιμο τμήμα· όλες οι διεργασίες πρέπει να έχουν την ίδια πιθανότητα να εξυπηρετηθούν μέσα σε κάποιο λογικό χρονικό διάστημα και όχι κάποιες να μονοπωλούν το σύστημα.

Οι δυο πρώτοι περιορισμοί προφυλάσσουν το σύστημα από το *αδιέξοδο* (deadlock) ή το *αμοιβαίο μπλοκάρισμα* (mutual blocking), όπου όλες οι διεργασίες εμποδίζονται να εκτελέσουν το κρίσιμο τμήμα τους ενώ τουλάχιστον μία θα μπορούσε να το κάνει.

Αμοιβαίος αποκλεισμός για δυο διεργασίες: η λύση του Peterson

Η πρώτη λύση που δόθηκε στο πρόβλημα του αμοιβαίου αποκλεισμού για δύο διεργασίες ήταν αυτή του Δανού μαθηματικού T. Dekker, όμως η λύση που έδωσε το 1981 ο J. Peterson είναι πολύ πιο απλή. Υπάρχει μια κοινή μεταβλητή με όνομα π.χ. seira, που δείχνει ποια από τις δυο διεργασίες έχει σειρά να εκτελέσει το κρίσιμο τμήμα της. Αν η μεταβλητή έχει την τιμή 1, τότε είναι σειρά της πρώτης διεργασίας, ενώ αν έχει την τιμή 2 είναι σειρά της δεύτερης. Επίσης υπάρχουν δύο μεταβλητές, μία για κάθε διεργασία, με ονόματα π.χ. eisodos1 και eisodos2. Η μεταβλητή eisodos1 έχει την τιμή true όταν η διεργασία 1 είναι έτοιμη να εκτελέσει ή εκτελεί το κρίσιμο τμήμα της, ενώ σε όλες τις άλλες φάσεις εκτέλεσής της έχει τιμή false.

Διεργασία 1

```
repeat
  A1 := DiabasePoso;
  eisodos1 := true;
  (* εκδήλωση πρόθεσης εισόδου
     στο κρίσιμο τμήμα *)
  seira := 2;

  (* Περίμενε να έρθει η σειρά σου *)
  while eisodos2 and seira <> 1 do
        do_nothing;

  Y := Y - A1; (* κρίσιμο τμήμα *)
  eisodos1 := false;
  DosePoso;
until false;
```

Διεργασία 2

```
repeat
  A2 := DiabasePoso;
  eisodos2 := true;
  (* εκδήλωση πρόθεσης εισόδου
     στο κρίσιμο τμήμα *)
  seira := 1;

  (* Περίμενε να έρθει η σειρά σου *)
  while eisodos1 and seira <> 2 do
            do_nothing;

  Y := Y - A2; (* κρίσιμο τμήμα *)
  eisodos2 := false;
  DosePoso;
until false;
```

Αντίστοιχες τιμές παίρνει και η μεταβλητή eisodos2, για τη διεργασία 2. Οι δύο διεργασίες έχουν τη μορφή που φαίνεται στα σχήματα.

Κάθε διεργασία περιμένει για την άλλη, εφόσον είναι η σειρά της άλλης και εκείνη έχει δηλώσει πρόθεση να εκτελέσει το κρίσιμο τμήμα της. Αν π.χ. είναι η σειρά της διεργασίας 2 αλλά εκείνη δεν είναι έτοιμη να εκτελέσει το κρίσιμο τμήμα της (δηλαδή eisodos2=false), τότε η διεργασία 1 μπαίνει στο δικό της κρίσιμο τμήμα, εμποδίζοντας τη 2 με τη βοήθεια της eisodos1. Αν και οι δυο διεργασίες θέλουν ταυτόχρονα να εκτελέσουν το κρίσιμο τμήμα τους, το εκτελεί πρώτα εκείνη που υποδεικνύεται από τη μεταβλητή seira.

Η λύση αυτή μπορεί να επεκταθεί και για την περίπτωση που έχουμε ν διεργασίες· η λύση αυτή όμως είναι αρκετά πολύπλοκη, ενώ το πρόβλημα λύνεται πολύ πιο απλά με τη βοήθεια των *σηματοφορέων* (semaphores) που περιγράφονται στη συνέχεια.

Ανακεφαλαίωση

Στα ταυτόχρονα προγράμματα η ύπαρξη δεδομένων που χρησιμοποιούν από κοινού πολλές διεργασίες (και τουλάχιστον μια από αυτές γράφει στα δεδομένα), εισάγει το ζήτημα της ακεραιότητας των δεδομένων αυτών. Το τμήμα κώδικα μιας διεργασίας το οποίο διαβάζει ή γράφει χρησιμοποιούν από κοινού χρησιμοποιούμενα δεδομένα ονομάζεται κρίσιμο τμήμα. Από μια ομάδα διεργασιών που χρησιμοποιούν από κοινού δεδομένα μόνο μια μπορεί να εκτελεί ανά πάσα στιγμή το κρίσιμο τμήμα της. Αν για κάποιο λόγο όλες οι διεργασίες της ομάδας εμποδίζονται να εκτελέσουν το κρίσιμο τμήμα τους, τότε έχουμε μια περίπτωση αδιεξόδου.

Γλωσσάριο όρων

Αδιέξοδο	*Deadlock*
Ακεραιότητα Δεδομένων	*Data Integrity*
Αμοιβαίο Μπλοκάρισμα	*Mutual Blocking*
Αμοιβαίος Αποκλεισμός	*Mutual Exclusion*
Δικαιοσύνη	*Fairness*
Κρίσιμο Τμήμα	*Critical Section*
Μοιραζόμενοι Πόροι	*Shared Resources*

Ερωτήσεις

? Τι είναι το κρίσιμο τμήμα; Περίγραψε το κρίσιμο τμήμα στο πρόβλημα των τραπεζικών αναλήψεων.

? Πότε έχουμε αμοιβαίο αποκλεισμό μεταξύ δύο διεργασιών;

? Εξήγησε τη λύση του Peterson για τον αμοιβαίο αποκλεισμό.

Μάθημα 7.5 Σηματοφορείς

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να παρουσιάσει τους σηματοφορείς και τον τρόπο που λύνουν το πρόβλημα του κρίσιμου τμήματος.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να ορίζεις τι είναι σηματοφορέας και οι λειτουργίες P και V*
- *Να περιγράφεις πώς λύνουν οι σηματοφορείς το πρόβλημα του κρίσιμου τμήματος*
- *Να απαριθμείς τις λίστες διεργασιών που κρατά το ΛΣ*
- *Να εξηγείς πώς λύνουμε απλά προβλήματα με σηματοφορείς*

Η λύση για το πρόβλημα του κρίσιμου τμήματος που είδαμε στο προηγούμενο μάθημα είναι σωστή, αλλά έχει δυο μειονεκτήματα:

1. **Προσθέτει πολλές μεταβλητές και επιπλέον εντολές στον κώδικα της διεργασίας κάνοντάς τον πολύπλοκο χωρίς λόγο**. Από τις 8 γραμμές κώδικα της κάθε διεργασίας που δόθηκαν στο προηγούμενο μάθημα, οι 3 μόνο είναι το πραγματικό πρόγραμμα, ενώ οι υπόλοιπες 5 είναι πρόσθετες.
2. **Δεν επεκτείνεται εύκολα για περισσότερες από δυο διεργασίες, όπου είναι ακόμα πιο πολύπλοκη**.

> Ο ανελκυστήρας μιας πολυκατοικίας χρησιμοποιείται με παρόμοιο τρόπο με τα μοιραζόμενα δεδομένα ενός ταυτόχρονου προγράμματος. Ο ένοικος που είναι μέσα στον ανελκυστήρα είναι η διεργασία που εκτελεί το κρίσιμο τμήμα της.
>
> Αν η χρήση του ανελκυστήρα γινόταν με τον τρόπο που περιγράψαμε στο προηγούμενο μάθημα, θα γινόταν μια πολύπλοκη διαδικασία: όποιος ήθελε να τον χρησιμοποιήσει θα δήλωνε αυτή του την πρόθεση, και ο ανελκυστήρας θα εξυπηρετούσε έναν-έναν όλους τους ενοίκους που τον χρειάζονται, σε κάθε όροφο με τη σειρά. Θα υπήρχαν κουμπιά για την εκδήλωση της ανάγκης χρήσης του ανελκυστήρα, για την επιλογή αυτού που έχει την προτεραιότητα, για την ελευθέρωση του ανελκυστήρα κλπ. Με αυτά θα γινόταν η συνεννόηση μεταξύ των ενοίκων κάθε φορά για το ποιος έχει προτεραιότητα, ποιος είναι επόμενος στη σειρά για να εξυπηρετηθεί και ποιος τελικά θα χρησιμοποιήσει τον ανελκυστήρα. Με μια τόσο πολύπλοκη και αργή διαδικασία μάλλον οι περισσότεροι θα χρησιμοποιούσαν το κλιμακοστάσιο!
>
> Τι γίνεται όμως στην πραγματικότητα; Όταν ο ανελκυστήρας χρησιμοποιείται, σε όλους τους ορόφους είναι αναμμένο το φωτάκι με την

ένδειξη «Κατειλημμένος». Όποιος θέλει να χρησιμοποιήσει τον ανελκυστήρα περιμένει μέχρι να σβήσει η φωτεινή ένδειξη και μετά εκδηλώνει την πρόθεσή του πατώντας ένα κουμπί. Όποιος προλάβει και το πατήσει πρώτος, καλεί τον ανελκυστήρα.

Οι λειτουργίες λοιπόν του συστήματος χρήσης του ανελκυστήρα είναι ουσιαστικά δυο:

- Ο ένοικος περιμένει να σβήσει η φωτεινή ένδειξη, πατά το κουμπί και καλεί τον ανελκυστήρα.
- Ο ένοικος βγαίνει από τον ανελκυστήρα και η φωτεινή ένδειξη σβήνει.

Η ιδέα της φωτεινής ένδειξης, που ενεργοποιείται με το πάτημα ενός κουμπιού, μπορεί να μεταφερθεί και στο πρόβλημα του κρίσιμου τμήματος, με τους *σηματοφορείς* (semaphores). Ο σηματοφορέας, που είναι ένας μετρητής με ακέραιες τιμές, αντιστοιχεί στη φωτεινή ένδειξη. Όταν έχει την τιμή 0, τότε η ένδειξη είναι αναμμένη, ενώ όταν έχει την τιμή 1 η ένδειξη είναι σβηστή. Το πάτημα του κουμπιού και η κλήση του ανελκυστήρα προκαλούν μείωση της μεταβλητής κατά 1, ενώ το σβήσιμο της φωτεινής ένδειξης αντιστοιχεί σε αύξηση της μεταβλητής κατά 1.

	Ανελκυστήρας	Σηματοφορέας
Λειτουργία P	Περίμενε να σβήσει η φωτεινή ένδειξη. Πάτα το κουμπί.	while s = 0 do_nothing; s := s - 1;
Λειτουργία V	Η φωτεινή ένδειξη σβήνει.	s := s + 1;

Οι σηματοφορείς και οι δυο λειτουργίες τους, με τα ονόματα P και V, παρουσιάστηκαν από το Dijkstra το 1965. Οι δυο λειτουργίες εκτελούνται αδιαίρετα, δηλαδή όταν στη λειτουργία P τελειώσει ο βρόχος «while s = 0 do_nothing», η επόμενη εντολή θα εκτελεστεί αμέσως χωρίς διακοπή από άλλη διεργασία. Αν δυο διεργασίες επιχειρήσουν ταυτόχρονα να εκτελέσουν τη λειτουργία P, τότε θα εκτελεστούν με τυχαία σειρά.

Διεργασία 1

```
repeat
  A1 := DiabasePoso;
     P(sema);
     (* κρίσιμο τμήμα *)
     Y := Y - A1;
     V(sema);
  DosePoso;
until false;
```

Διεργασία 2

```
repeat
  A2 := DiabasePoso;
     P(sema);
     (* κρίσιμο τμήμα *)

     Y := Y - A2;
     V(sema);
  DosePoso;
until false;
```

Το πρόβλημα των αναλήψεων λύνεται τώρα με ένα σηματοφορέα, με όνομα π.χ. «sema», όπως φαίνεται στο σχήμα.

Υλοποίηση των λειτουργιών P και V

Όπως είδαμε, ένας σηματοφορέας είναι πρακτικά ένας μετρητής, τον οποίο μοιράζονται όσες διεργασίες χρησιμοποιούν το σηματοφορέα. Για να γίνονται σωστά οι λειτουργίες στο σηματοφορέα, πρέπει η πρόσβαση στο μοιραζόμενο αυτό μετρητή να γίνεται με ασφάλεια (κρίσιμο τμήμα). Αυτός είναι ο λόγος που οι λειτουργίες P και V πρέπει να εκτελούνται αδιαίρετα.

Σε όλους τους υπολογιστές οι εντολές σε γλώσσα μηχανής, δηλαδή στο στοιχειωδέστερο επίπεδο προγραμματισμού τους, εκτελούνται αδιαίρετα. Έτσι αν οι δυο λειτουργίες υλοποιούνται σαν εντολές γλώσσας μηχανής, τότε ικανοποιείται αυτόματα ο περιορισμός της αδιαίρετης εκτέλεσής τους. Αυτό όμως μπορεί να δημιουργήσει δυο σοβαρά προβλήματα:

- Αν ο σηματοφορέας είναι ήδη κατειλημμένος από μια διεργασία, δηλαδή έχει τιμή 0, η επόμενη που θα εκτελέσει τη λειτουργία P θα αρχίσει να εκτελεί *αδιαίρετα* το βρόχο «while s = 0 do_nothing». Αφού η εκτέλεση είναι αδιαίρετη, όσο ο βρόχος εκτελείται δεν είναι δυνατόν να διακοπεί η διεργασία για να εκτελεστεί κάποια άλλη, συνεπώς και να απελευθερωθεί ο σηματοφορέας από τη διεργασία που τον κατέχει. Ο υπολογιστής τότε βρίσκεται σε κατάσταση αδιεξόδου και εκτελεί επ' άπειρο το βρόχο.

- Ακόμα και αν το προηγούμενο πρόβλημα ξεπεραστεί, κάθε εκτέλεση της λειτουργίας P από μια διεργασία θα συνεπάγεται τη σπατάλη αρκετού υπολογιστικού χρόνου στην εκτέλεση του βρόχου. Οι διεργασίες, σε κάθε κβάντο χρόνου που τους αντιστοιχεί θα εκτελούν το βρόχο, ξοδεύοντας έτσι χρόνο που θα μπορούσε να χρησιμοποιηθεί πιο «παραγωγικά».

Για να ξεπεραστούν τα προβλήματα αυτά, οι λειτουργίες P και V δεν υλοποιούνται στην πραγματικότητα όπως ορίζονται. Για την υλοποίηση των σηματοφορέων χρησιμοποιούνται *λίστες αναμονής* (waiting lists).

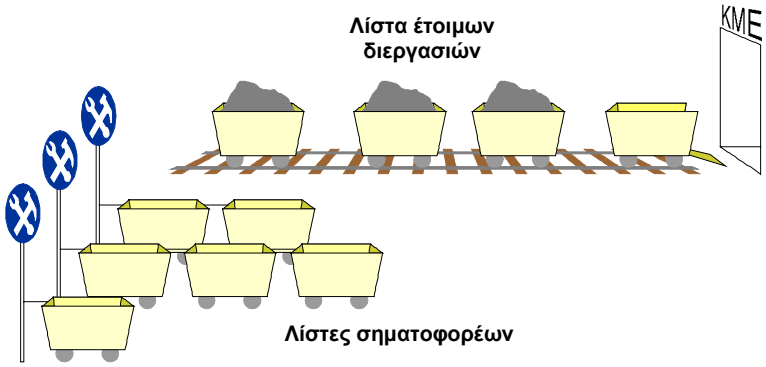
Κάθε σηματοφορέας έχει τη δική του λίστα αναμονής στην οποία περιμένουν οι διεργασίες που χρειάζονται το σηματοφορέα, για όσο χρόνο αυτός είναι κατειλημμένος. Όταν το ΛΣ επιλέγει την επόμενη διεργασία που θα εκτελεστεί, δε λαμβάνει υπόψη αυτές που περιμένουν στη λίστα κάποιου σηματοφορέα, γιατί έτσι και αλλιώς αυτή δεν έχει τίποτα να κάνει μέχρι να ελευθερωθεί ο σηματοφορέας.

Η λειτουργία P λοιπόν γίνεται ως εξής: αν ο σηματοφορέας είναι ελεύθερος, τότε η διεργασία τον καταλαμβάνει. Αν δεν είναι ελεύθερος, η διεργασία καταγράφεται στη λίστα αναμονής του σηματοφορέα, όπου και παραμένει χωρίς να εκτελείται. Στη λειτουργία V, όπου μια διεργασία ελευθερώνει το σηματοφορέα, το ΛΣ εξετάζει τη λίστα αναμονής του. Επιλέγει από αυτή την πρώτη διεργασία και τη διαγράφει από τη λίστα αναμονής, εισάγοντάς τη στη λίστα των διεργασιών που εκτελούνται εκ περιτροπής στην ΚΜΕ. Σύντομα θα έλθει η σειρά της διεργασίας αυτής να εκτελεστεί, και τότε θα καταλάβει το σηματοφορέα ολοκληρώνοντας έτσι τη λειτουργία P.

Η λίστα αναμονής ενός σηματοφορέα οργανώνεται σαν *ουρά* (queue). Σε μια ουρά, όπως π.χ. στο ταμείο μιας τράπεζας, επιλέγεται από την ουρά η διεργασία που περιμένει την περισσότερη ώρα.

Οι λίστες του λειτουργικού συστήματος

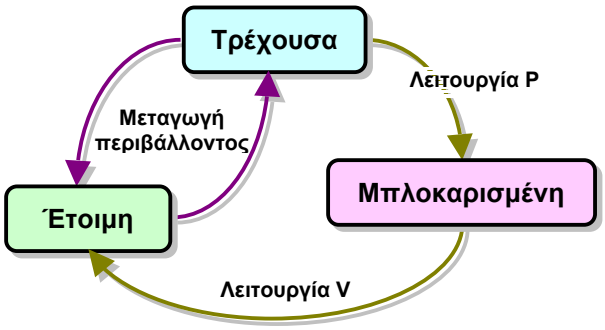
Με τη μέθοδο των λιστών αναμονής, οι διεργασίες χωρίζονται σε κατηγορίες: Για κάθε σηματοφορέα έχουμε την κατηγορία των διεργασιών που περιμένουν γι' αυτόν, και υπάρχει η κατηγορία των διεργασιών που δεν περιμένουν για κανένα σηματοφορέα. Για να μπορεί το ΛΣ να βρίσκει εύκολα και γρήγορα ποιες διεργασίες δεν περιμένουν σε κάποιο σηματοφορέα, κρατά μια ξεχωριστή λίστα με αυτές. Η λίστα αυτή ονομάζεται *λίστα έτοιμων διεργασιών* (ready list, RL)· από αυτήν επιλέγει το ΛΣ την επόμενη διεργασία που θα εκτελεστεί. Συνήθως και η λίστα έτοιμων διεργασιών είναι οργανωμένη σαν ουρά.



Ο τρόπος λειτουργίας της ΚΜΕ είναι παρόμοιος με αυτόν ενός ορυχείου, όπου βαγονάκια μεταφέρουν από τις σήραγγες το μετάλλευμα στο εργοστάσιο για επεξεργασία. Τα βαγονάκια - διεργασίες μπαίνουν ένα-ένα στο εργοστάσιο- ΚΜΕ από μια σιδηροδρομική γραμμή - λίστα έτοιμων διεργασιών για να εξυπηρετηθούν. Όμως κατά τη διάρκεια της λειτουργίας του ορυχείου τα βαγονάκια χρειάζονται συντήρηση· άλλο χρειάζεται λάδωμα στους τροχούς, άλλο επισκευή στα φρένα, και άλλο χρειάζεται επιδιόρθωση στον καταπέλτη. Ανάλογα με την επισκευή που χρειάζονται μπαίνουν στην ουρά του αντίστοιχου συνεργείου-σηματοφορέα, και όταν έρθει η σειρά τους επισκευάζονται. Για όσο

διάστημα βρίσκονται υπό επισκευή δε χρησιμοποιούνται, οπότε βρίσκονται εκτός της γραμμής-λίστας έτοιμων διεργασιών.

Μια διεργασία μπορεί λοιπόν να βρίσκεται στις εξής *καταστάσεις* (process states):



- εκτελείται από την ΚΜΕ: η διεργασία ονομάζεται *τρέχουσα* (running)
- περιμένει για εκτέλεση στη λίστα έτοιμων διεργασιών: η διεργασία ονομάζεται *έτοιμη* (ready)
- περιμένει στην ουρά κάποιου σηματοφορέα: η διεργασία είναι *μπλοκαρισμένη* (blocked)

Στο σχήμα βλέπουμε τον τρόπο που μία διεργασία μπορεί να μεταβεί από μία κατάσταση σε άλλη. Η μετάβαση από την κατάσταση «Τρέχουσα» στην κατάσταση «Μπλοκαρισμένη» με τη λειτουργία P γίνεται μόνο εφόσον ο σηματοφορέας έχει την τιμή 0.

Στη συνέχεια θα συναντήσουμε και άλλες καταστάσεις στις οποίες μπορεί να βρίσκεται μια διεργασία.

Το πρόβλημα των αναγνωστών και συγγραφέων

Στις *βάσεις δεδομένων* (databases) παρουσιάζεται ένα πολύ συνηθισμένο πρόβλημα αμοιβαίου αποκλεισμού διεργασιών, το πρόβλημα των αναγνωστών και συγγραφέων. Μια ομάδα διεργασιών μοιράζεται ένα αρχείο, στο οποίο άλλες διεργασίες θέλουν να γράψουν και να το τροποποιήσουν, ενώ άλλες θέλουν μόνο να διαβάσουν από αυτό. Για να εξασφαλίζεται το ότι τα περιεχόμενα του αρχείου είναι σωστά, πρέπει οι συγγραφείς να έχουν αποκλειστική πρόσβαση στο αρχείο. Όταν γράφει λοιπόν μια διεργασία-συγγραφέας καμία άλλη διεργασία δεν έχει πρόσβαση στο αρχείο. Αντίθετα, πολλές διεργασίες-αναγνώστες μπορούν να διαβάζουν ταυτόχρονα.

Για να δώσουμε μια λύση στο πρόβλημα αυτό πρέπει να αποφασίσουμε ποιοι θα έχουν προτεραιότητα: οι αναγνώστες ή οι συγγραφείς, όταν ταυτόχρονα μια διεργασία θέλει να διαβάσει και μια θέλει να γράψει. Εμείς θα υποθέσουμε εδώ ότι προηγούνται οι αναγνώστες, δηλαδή οι συγγραφείς περιμένουν να ολοκληρωθούν οι λειτουργίες ανάγνωσης.

- Στη συνέχεια πρέπει να αποφασίσουμε ποιους σηματοφορείς και μεταβλητές θα χρησιμοποιήσουμε και ποιο ρόλο θα αναλάβει ο καθένας. Θα εξετάζουμε ένα-ένα τα προβλήματα ακεραιότητας δεδομένων που υπάρχουν και θα προσθέτουμε διαδοχικά εντολές στον κώδικα των διεργασιών. Οι απλούστερες μορφές του κώδικα φαίνονται στο σχήμα. Η διεργασία - συγγραφέας γράφει στο αρχείο myfile, ενώ η διεργασία - αναγνώστης διαβάζει από αυτό.

Συγγραφείς

```
write(myfile, "…");
```

Αναγνώστες

```
read(myfile, mytext);
```

- Οπωσδήποτε πρέπει να χρησιμοποιήσουμε ένα σηματοφορέα ο οποίος θα εξασφαλίζει το ότι μόνο ένας συγγραφέας θα γράφει ανά πάσα στιγμή στο αρχείο, ώστε οι αλλαγές στο αρχείο να γίνονται σωστά. Κάθε συγγραφέας προτού γράψει θα καταλαμβάνει το σηματοφορέα αυτό και όταν τελειώνει θα τον ελευθερώνει. Το όνομα του σηματοφορέα θα είναι write_sema.
  Τώρα κάθε διεργασία - συγγραφέας εκτελεί μία λειτουργία P στο σηματοφορέα, και όταν αυτή ολοκληρωθεί εκτελεί την εγγραφή. Στη συνέχεια απελευθερώνει το σηματοφορέα με τη λειτουργία V για να τον διαθέσει στους υπόλοιπους συγγραφείς.

Συγγραφείς

```
P(write_sema);

write(myfile, "…");

V(write_sema);
```

- Για να βεβαιωθούν οι αναγνώστες ότι δε γράφει κάποιος στο αρχείο πριν διαβάσουν, πρέπει πριν από την ανάγνωση να καταλάβουν το σηματοφορέα write_sema. Όμως, όταν ένας αναγνώστης έχει καταλάβει το σηματοφορέα με τη λειτουργία P, όλοι οι άλλοι που θα εκτελέσουν επίσης την P θα τον περιμένουν. Αυτό είναι κάτι ανεπιθύμητο, μια που πολλοί αναγνώστες μπορούν να διαβάζουν ταυτόχρονα. Έτσι, αρκεί ο *πρώτος* αναγνώστης να καταλάβει το σηματοφορέα αυτό, και οι υπόλοιποι μπορούν πλέον να διαβάζουν άφοβα. Ο τελευταίος αναγνώστης αντίστοιχα πρέπει να ελευθερώσει τον write_sema.
  Για να διαπιστώσει μια διεργασία - αναγνώστης αν είναι η πρώτη, πρέπει σε μια μεταβλητή να κρατείται το πλήθος των αναγνωστών που είναι ενεργοί ανά πάσα στιγμή· στην μεταβλητή θα δώσουμε το όνομα readcount. Αρχικά η μεταβλητή έχει την τιμή 0, κάθε αναγνώστης την αυξάνει πριν από την ανάγνωση, και τη μειώνει μετά από αυτήν. Ο πρώτος αναγνώστης, αφού αυξήσει τη readcount, θα διαπιστώσει ότι έχει την τιμή 1 και θα καταλάβει το σηματοφορέα write_sema. Ο τελευταίος, όταν τη μειώσει, θα διαπιστώσει ότι έχει πάλι τιμή 0 και θα ελευθερώσει το σηματοφορέα write_sema.

Αναγνώστες

```
incr(readcount);
if readcount=1 then
   P(write_sema);

read(myfile, mytext);

decr(readcount);
if readcount=0 then
   V(write_sema);
```

- Όμως η προσθήκη που κάναμε στους αναγνώστες έχει ένα πρόβλημα: μπορεί πολλοί αναγνώστες μαζί να αυξάνουν ή να μειώνουν τη μεταβλητή readcount, και μετά να ελέγχουν την τιμή της. Σε μια τέτοια περίπτωση τα αποτελέσματα είναι ανεξέλεγκτα και συνήθως όχι σωστά. Μια που η readcount είναι μοιραζόμενη μεταβλητή, η πρόσβαση σε αυτήν πρέπει να φυλάσσεται από ένα σηματοφορέα, στον οποίο θα δώσουμε το όνομα rc_sema. Οι αναγνώστες θα καταλαμβάνουν τον rc_sema πριν να αυξήσουν την τιμή της readcount και θα τον ελευθερώνουν *αφού* έχουν ελέγξει την τιμή της.

Αναγνώστες

```
P(rc_sema);
incr(readcount);
if readcount=1 then
   P(write_sema);
V(rc_sema);

read(myfile, mytext);

P(rc_sema);
decr(readcount);
if readcount=0 then
   V(write_sema);
V(rc_sema);
```

Ανακεφαλαίωση

Οι σηματοφορείς είναι ένας μηχανισμός που λύνει με απλό και λειτουργικό τρόπο το πρόβλημα του κρίσιμου τμήματος. Είναι πρακτικά ακέραιες μεταβλητές, στις οποίες επενεργούν δυο λειτουργίες: η λειτουργία P, που πρέπει να εκτελεστεί πριν από την εκτέλεση του κρίσιμου τμήματος, και η λειτουργία V που πρέπει να εκτελεστεί αμέσως μετά. Η υλοποίηση των σηματοφορέων στα πραγματικά ΛΣ γίνεται με λίστες αναμονής όπου περιμένουν οι διεργασίες για να καταλάβουν το σηματοφορέα. Μια διεργασία περιμένει στην ουρά ενός σηματοφορέα, εκτελείται από την ΚΜΕ ή είναι έτοιμη για εκτέλεση.

Γλωσσάριο όρων

Έτοιμη Διεργασία	*Ready Process*
Κατάσταση Διεργασίας	*Process State*
Λειτουργία P	
Λειτουργία V	
Λίστα Αναμονής	*Waiting List*
Λίστα Έτοιμων Διεργασιών	*Ready List*
Μπλοκαρισμένη Διεργασία	*Blocked Process*
Ουρά	*Queue*
Σηματοφορέας	*Semaphore*
Τρέχουσα Διεργασία	*Running Process*

Ερωτήσεις

? Τι πλεονεκτήματα έχει ένας σηματοφορέας όταν θέλουμε να λύσουμε το πρόβλημα του κρίσιμου τμήματος για πολλές διεργασίες;

? Γιατί υλοποιούμε τους σηματοφορείς με λίστες αναμονής και όχι με τον τρόπο που παρουσιάσαμε στη θεωρία;

? Πόσους αναγνώστες και συγγραφείς μπορούμε να έχουμε στο παράδειγμα;

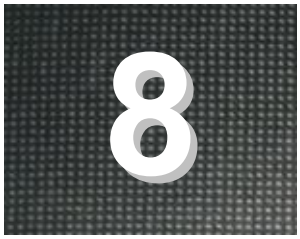
Τι μάθαμε σε αυτό το κεφάλαιο

- Το Λειτουργικό Σύστημα είναι ένα σύνολο προγραμμάτων που οργανώνουν και επιβλέπουν τις λειτουργίες του υλικού σε ένα υπολογιστικό σύστημα.
- Τα προγράμματα που βρίσκονται σε κάποιο στάδιο της εκτέλεσής τους από την ΚΜΕ ονομάζονται διεργασίες.
- Οι βασικοί τρόποι απεικόνισης διεργασιών είναι ο γράφος προήγησης και οι εντολές parbegin και parend.
- Το τμήμα μιας διεργασίας που χρησιμοποιεί μοιραζόμενα δεδομένα είναι το κρίσιμο τμήμα της. Μόνο μία διεργασία μπορεί να εκτελεί το κρίσιμο τμήμα της κάθε φορά.
- Οι σηματοφορείς είναι μία πολύ απλή και λειτουργική λύση για το πρόβλημα του κρίσιμου τμήματος.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Ν. Μπιλάλη, Π. Τσανάκα, Λειτουργικά Συστήματα: Αρχές Λειτουργίας, Συμμετρία, 1989.
- Silberschatz A., Peterson J., Galvin P., Operating System Concepts, Addison - Wesley, 1991.

Κεφάλαιο



Διαχείριση ΚΜΕ στα Λειτουργικά Συστήματα

Σκοπός του κεφαλαίου αυτού είναι να σου γνωρίσει τον τρόπο με τον οποίο ένα λειτουργικό σύστημα διαχειρίζεται τις διεργασίες του, με τέτοιο τρόπο ώστε να εξυπηρετούνται όλες οι ανάγκες των χρηστών του με ικανοποιητικό τρόπο.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να γνωρίζεις πώς εναλλάσσονται οι διεργασίες στην ΚΜΕ από το χρονοδρομολογητή.
- Να γνωρίζεις τα διάφορα είδη και τους αλγορίθμους χρονοδρομολόγησης.
- Να εφαρμόζεις τους αλγορίθμους χρονοδρομολόγησης σε πρακτικά παραδείγματα και να υπολογίζεις διάφορους δείκτες απόδοσης.

Μαθήματα

8.1 Χρονοδρομολόγηση

8.2 Αλγόριθμοι Χρονοδρομολόγησης

Μάθημα 8.1 Χρονοδρομολόγηση

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να εξηγήσει το ρόλο και τη λειτουργία της χρονοδρομολόγησης σε ένα λειτουργικό σύστημα.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να ορίζεις τη χρονοδρομολόγηση της ΚΜΕ*
- *Να απαριθμείς τα επίπεδα χρονοδρομολόγησης που υπάρχουν*
- *Να περιγράφεις τον κύκλο εκτέλεσης των διεργασιών*

Για να εκτελούνται οι διάφορες διεργασίες και να παράγεται υπολογιστικό έργο, πρέπει πρώτα απ' όλα το ΛΣ να αποφασίζει ποια διεργασία θα εκτελείται ανά πάσα στιγμή στην ΚΜΕ του υπολογιστή.

Η ενότητα του ΛΣ που λαμβάνει διάφορες αποφάσεις σχετικά με την ανάθεση της ΚΜΕ στις διάφορες διεργασίες ονομάζεται *χρονοδρομολόγηση ΚΜΕ* (CPU scheduling).

Όταν σχεδιάζεται ένα ΛΣ, η επιλογή των μεθόδων που αυτό θα χρησιμοποιεί για τη χρονοδρομολόγηση της ΚΜΕ είναι πολύ σημαντική για την καλή απόδοση του συστήματος. Ανάλογα με την οργάνωση του υλικού και τις υπολογιστικές ανάγκες που το σύστημα (υλικό + λογισμικό) θα καλύψει, μπορούν να ακολουθηθούν διάφορες μεθοδολογίες.

Η χρονοδρομολόγηση της ΚΜΕ αντιστοιχεί με την επιλογή από το ζαχαροπλάστη του γλυκού που παρασκευάζει κάθε φορά. Επειδή γενικά οι παραγγελίες είναι πολλές, δεν τις εκτελεί όλες ταυτόχρονα. Επιλέγει μια ομάδα από αυτές, και τις εκτελεί μαζί· κάθε φορά που μια από αυτές ολοκληρώνεται, την αντικαθιστά με μια νέα. Ανάλογα με τις ανάγκες των πελατών, αλλά και τις δυνατότητες που του παρέχει η κουζίνα του, ο ζαχαροπλάστης πρέπει να λάβει διάφορες αποφάσεις:

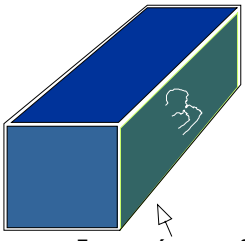
- Πόσος χρόνος θα διατίθεται για κάθε συνταγή μέχρι αυτή να διακοπεί για να πάρει κάποια άλλη τη θέση της;
- Ποια συνταγή από αυτές που μπορούν να συνεχιστούν θα επιλέξει κάθε φορά ο ζαχαροπλάστης για να συνεχίσει;
- Ποιες από όλες τις διαθέσιμες παραγγελίες για γλυκά θα επιλέξει για να αρχίσει να παρασκευάζει ταυτόχρονα;

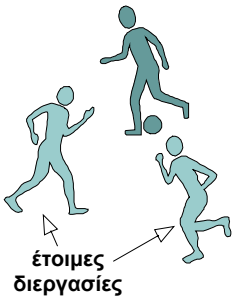
- Πότε πρέπει να αλλάξει την ομάδα των συνταγών που εκτελεί ταυτόχρονα, έτσι ώστε να πετύχει την καλύτερη δυνατή εκμετάλλευση του χρόνου και των σκευών του;

Βλέπουμε λοιπόν ότι η χρονοδρομολόγηση της ΚΜΕ γίνεται σε δυο επίπεδα:

1. Στο πρώτο επίπεδο, οι διεργασίες που πρέπει να εκτελεστούν βρίσκονται αποθηκευμένες όλες μαζί σε κάποια μονάδα μαζικής αποθήκευσης (π.χ. το σκληρό δίσκο). Από αυτές επιλέγεται μια ομάδα διεργασιών, οι οποίες θα εναλλάσσονται εκτελούμενες στην ΚΜΕ.
2. Στο δεύτερο επίπεδο βρίσκονται μόνο οι διεργασίες οι οποίες έχουν επιλεγεί από το πρώτο. Οι διεργασίες αυτές βρίσκονται πλέον στην κύρια μνήμη, και είναι καταχωρημένες στη λίστα έτοιμων διεργασιών. Η χρονοδρομολόγηση πλέον περιλαμβάνει την επιλογή σε τακτά χρονικά διαστήματα μιας νέας διεργασίας για να εκτελεστεί από την ΚΜΕ.

Τα δυο επίπεδα χρονοδρομολόγησης της ΚΜΕ μπορούν να αντιστοιχιστούν με έναν ποδοσφαιρικό αγώνα. Αρχικά υπάρχουν δυο δεκαεξάδες παικτών, από τις οποίες όμως οι προπονητές επιλέγουν δυο ενδεκάδες για να αγωνιστούν. Η επιλογή αυτή αντιστοιχεί στο πρώτο στάδιο χρονοδρομολόγησης, όπου κάποιες διεργασίες επιλέγονται για να παίξουν ενεργό ρόλο, ενώ οι υπόλοιπες παραμένουν «ανενεργές». Το δεύτερο στάδιο χρονοδρομολόγησης αντιστοιχεί στην εναλλαγή της μπάλας μεταξύ των παικτών (όπου η μπάλα παριστάνει την ΚΜΕ)· μόνο ένας παίκτης μπορεί να έχει στην κατοχή του τη μπάλα, και μάλιστα ένας από τους εικοσιδύο ενεργούς παίκτες που συμμετέχουν στο παιγνίδι. Οι αναπληρωματικοί πρέπει να μπουν πρώτα στον αγώνα για να αποκτήσουν τη δυνατότητα να πάρουν τη μπάλα.





Είδη χρονοδρομολόγησης

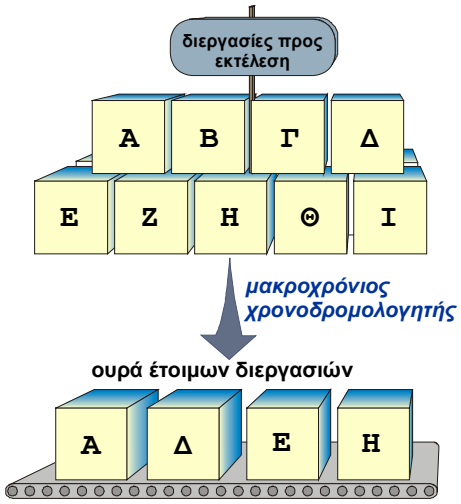
Τα δυο αυτά επίπεδα χρονοδρομολόγησης γίνονται ανεξάρτητα· υπάρχουν λοιπόν στο ΛΣ δυο *χρονοδρομολογητές* (schedulers), ένας για κάθε επίπεδο.

- Ο *μακροχρόνιος χρονοδρομολογητής* ή *χρονοδρομολογητής εργασιών* (long term scheduler ή job scheduler), καθορίζει ποιες διεργασίες (προγράμματα έτοιμα να εκτελεστούν) εισέρχονται στο σύστημα για εκτέλεση. Από όλες τις διεργασίες που έχουν υποβληθεί από τους χρήστες για εκτέλεση, και βρίσκονται σε κάποια μονάδα μαζικής αποθήκευσης, ο μακροχρόνιος χρονοδρομολογητής επιλέγει ορισμένες και τις φορτώνει στη μνήμη για επεξεργασία.
- Ο *βραχυχρόνιος χρονοδρομολογητής* ή *χρονοδρομολογητής ΚΜΕ* (short term scheduler ή CPU scheduler), επιλέγει για να εκτελέσει μια από τις διεργασίες οι οποίες είναι φορτωμένες στη μνήμη και είναι έτοιμες να χρησιμοποιήσουν την ΚΜΕ. Οι διεργασίες αναμένουν στη λίστα έτοιμων διεργασιών (τη λίστα του ΛΣ με τις διεργασίες που μπορούν να εκτελεστούν από την ΚΜΕ), από όπου κάνει την επιλογή του ο βραχυχρόνιος χρονοδρομολογητής.

Η κύρια διαφοροποίηση μεταξύ των δυο αυτών χρονοδρομολογητών είναι η συχνότητα με την οποία εκτελούνται. Ο μακροχρόνιος χρονοδρομολογητής εκτελείται σε αραιά χρονικά διαστήματα, ενώ αντίθετα ο βραχυχρόνιος εκτελείται πολύ συχνά

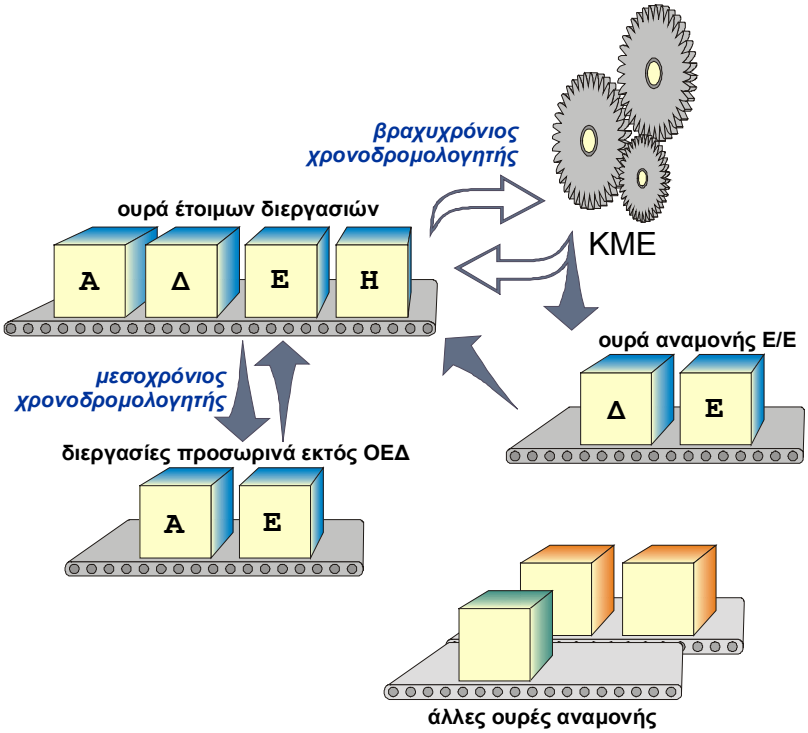
(π.χ. κάθε 10 msec). Για να μην επιβαρύνει λοιπόν το σύστημα, ο βραχυχρόνιος χρονοδρομολογητής πρέπει να είναι πολύ γρήγορος στην επιλογή της διεργασίας που θα εκτελεστεί.

Εκτός από τους δυο αυτούς χρονοδρομολογητές, πολλές φορές υπάρχει και ένα ενδιάμεσο επίπεδο χρονοδρομολόγησης, το οποίο αναλαμβάνει ο *μεσοχρόνιος χρονοδρομολογητής* (mid term scheduler). Αυτός ελέγχει σε τακτά χρονικά διαστήματα τη λίστα έτοιμων διεργασιών, και επιλέγει μερικές φορές από αυτή κάποιες διεργασίες οι οποίες διακόπτονται προσωρινά. Σε άλλες περιπτώσεις ενεργοποιεί ξανά κάποιες διεργασίες που είχε διακόψει παλαιότερα και τις επιστρέφει στη λίστα έτοιμων διεργασιών για εκτέλεση. Με τον τρόπο αυτό, επιδιώκεται να είναι ομοιόμορφα κατανεμημένο το υπολογιστικό φορτίο κατά τη λειτουργία του υπολογιστή και ο χρόνος απόκρισης να μην ξεφεύγει από κάποια όρια αποδεκτής λειτουργίας.



Στα σχήματα παριστάνονται τα διάφορα στάδια της χρονοδρομολόγησης από τα οποία περνά μια διεργασία. Η είσοδός της στην ουρά των έτοιμων διεργασιών γίνεται από το μακροχρόνιο χρονοδρομολογητή, και η μετακίνησή της από και προς την ΚΜΕ από το βραχυχρόνιο. Ο μεσοχρόνιος χρονοδρομολογητής μετακινεί διεργασίες προς και από μια άλλη ουρά διεργασιών, οι οποίες βρίσκονται προσωρινά εκτός του ανταγωνισμού για χρήση της ΚΜΕ.

Όταν μια διεργασία αφήνει την ΚΜΕ μπορεί να επιστρέψει στη λίστα έτοιμων διεργασιών, αλλά υπάρχει και η περίπτωση να βρεθεί σε μια ουρά αναμονής, π.χ. για είσοδο/έξοδο δεδομένων από κάποια συσκευή. Η *είσοδος/έξοδος* -αλλιώς Ε/Ε - *δεδομένων* (data input/output, I/O) περιλαμβάνει λειτουργίες όπως ανάγνωση από το σκληρό δίσκο, αποστολή δεδομένων στον εκτυπωτή μέσω της παράλληλης θύρας, αποστολή δεδομένων από την κάρτα δικτύου. Γενικά η διεκπεραίωση Ε/Ε είναι πολύ πιο αργή από την εκτέλεση εντολών της ΚΜΕ· κατά τη διάρκεια μιας μόνο ανάγνωσης από το σκληρό δίσκο μπορούν να εκτελεστούν χιλιάδες εντολές από την ΚΜΕ. Όσο λοιπόν μια διεργασία αναμένει την ολοκλήρωση μιας λειτουργίας Ε/Ε μπορεί να παραχωρήσει την ΚΜΕ σε άλλες, και αυτή περιμένει σε μια ουρά την ολοκλήρωση της λειτουργίας Ε/Ε. Η παράλληλη εκτέλεση Ε/Ε από μια διεργασία και εκτέλεση στην ΚΜΕ από μια άλλη βελτιώνει την απόδοση του συστήματος.



Κύκλος εκτέλεσης διεργασιών

Η ζωή μιας διεργασίας είναι ένας συνεχής κύκλος από εκτέλεση στην ΚΜΕ και αναμονή για ολοκλήρωση μιας λειτουργίας Ε/Ε. Κάθε συνεχόμενη περίοδος εκτέλεσης στην ΚΜΕ ονομάζεται *έκρηξη ΚΜΕ* (CPU burst) ενώ μια συνεχόμενη περίοδος αναμονής για Ε/Ε ονομάζεται *έκρηξη Ε/Ε* (I/O burst). Η ζωή τής διεργασίας λοιπόν

ξεκινά με μια έκρηξη ΚΜΕ, η οποία ακολουθείται από μια έκρηξη Ε/Ε, κλπ. Ολοκληρώνεται με μια τελευταία έκρηξη ΚΜΕ κατά την οποία η διεργασία τερματίζεται.

Αν και οι διάρκειες των εκρήξεων διαφέρουν πολύ μεταξύ διαφορετικών υπολογιστών ή διεργασιών, γενικά σε κάθε διεργασία παρατηρείται ένας μεγάλος αριθμός από σύντομες εκρήξεις ΚΜΕ και ένας μικρός αριθμός από πολύ μεγάλες.

Οι διεργασίες μπορούν να κατηγοριοποιηθούν ανάλογα με το συνολικό χρόνο που περνούν σε εκρήξεις του ενός ή του άλλου είδους. Έτσι έχουμε τις διεργασίες που είναι *προσανατολισμένες στην Ε/Ε* (I/O bound) και περνούν τον περισσότερο χρόνο τους σε εκρήξεις Ε/Ε, και αυτές που είναι *προσανατολισμένες στην ΚΜΕ* (CPU bound) και απασχολούν περισσότερο την ΚΜΕ.

Ο μακροχρόνιος χρονοδρομολογητής πρέπει να κάνει ιδιαίτερα προσεκτική επιλογή των διεργασιών που θα εισαγάγει στο σύστημα για εκτέλεση. Αν διαλέξει πολλές διεργασίες προσανατολισμένες σε Ε/Ε τότε η ΚΜΕ θα υποαπασχολείται. Αν επιλέξει πολλές διεργασίες προσανατολισμένες στην ΚΜΕ, τότε οι συσκευές Ε/Ε θα μένουν ανενεργές ενώ η ΚΜΕ θα απασχολείται ασταμάτητα.

Ανακεφαλαίωση

Η χρονοδρομολόγηση σε ένα ΛΣ αποφασίζει ποια διεργασία θα χρησιμοποιεί ανά πάσα στιγμή την ΚΜΕ. Συνήθως σε ένα ΛΣ υπάρχει ο μακροχρόνιος χρονοδρομολογητής, ο οποίος επιλέγει μια ομάδα διεργασιών για να εκτελεστούν ταυτόχρονα, και ο βραχυχρόνιος χρονοδρομολογητής που επιλέγει μια συγκεκριμένη διεργασία και της αναθέτει την ΚΜΕ. Ανάμεσα σε αυτούς τους δυο μπορεί να παρεμβληθεί και ο μεσοχρόνιος χρονοδρομολογητής. Μια διεργασία μπορεί να εκτελείται στην ΚΜΕ ή να περιμένει για κάποια λειτουργία Ε/Ε. Οι χρονοδρομολογητές πρέπει να εξασφαλίζουν ότι θα υπάρχει ισορροπία ανάμεσα στις διεργασίες που απαιτούν πολύ χρόνο ΚΜΕ και εκείνες που απαιτούν πολύ χρόνο Ε/Ε έτσι ώστε το σύστημα να έχει καλή απόδοση και χρησιμοποίηση τόσο της ΚΜΕ όσο και των συσκευών Ε/Ε.

Γλωσσάριο όρων

Βραχυχρόνιος Χρονοδρομολογητής	*Short Term Scheduler*
Διεργασία Προσανατολισμένη σε Ε/Ε	*I/O Bound Process*
Διεργασία Προσανατολισμένη στην ΚΜΕ	*CPU Bound Process*
Είσοδος/Έξοδος Δεδομένων - Ε/Ε	*Data Input/Output - I/O*
Έκρηξη Ε/Ε	*I/O Burst*
Έκρηξη ΚΜΕ	*CPU Burst*
Μακροχρόνιος Χρονοδρομολογητής	*Long Term Scheduler*
Μεσοχρόνιος Χρονοδρομολογητής	*Mid Term Scheduler*
Χρονοδρομολόγηση ΚΜΕ	*CPU Scheduling*
Χρονοδρομολογητής	*Scheduler*
Χρονοδρομολογητής Εργασιών	*Job Scheduler*
Χρονοδρομολογητής ΚΜΕ	*CPU Scheduler*

Ερωτήσεις

? Ποιο είναι το έργο του χρονοδρομολογητή σε ένα ΛΣ; Νομίζεις ότι είναι σημαντικό;

? Πόσα και ποια επίπεδα χρονοδρομολόγησης υπάρχουν; Είναι όλα τους απαραίτητα;

? Τι είναι η «έκρηξη ΚΜΕ»; Πόσες εκρήξεις ΚΜΕ μπορεί να έχει μία διεργασία;

Μάθημα 8.2 Αλγόριθμοι Χρονοδρομολόγησης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει το ρόλο και την αξιολόγηση των αλγορίθμων χρονοδρομολόγησης, και να παρουσιάσει τους κυριότερους τέτοιους αλγορίθμους.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να εξηγείς τη λειτουργία των αλγορίθμων χρονοδρομολόγησης*
- *Να περιγράφεις πώς αξιολογούμε τους αλγορίθμους χρονοδρομολόγησης*
- *Να ορίζεις τι είναι διακοπτός και μη διακοπτός αλγόριθμος*
- *Να απαριθμείς τους κυριότερους μη διακοπτούς αλγόριθμους χρονοδρομολόγησης*
- *Να απαριθμείς τους κυριότερους διακοπτούς αλγόριθμους χρονοδρομολόγησης*

Στο μάθημα 7.2 είδαμε ότι οι διεργασίες μπορούν να μοιράζονται μεταξύ τους την ΚΜΕ. Στο μάθημα αυτό θα δούμε τους τρόπους που μπορεί να γίνει αυτό, εξετάζοντας τους διάφορους αλγορίθμους χρονοδρομολόγησης.

Η μεθοδολογία την οποία χρησιμοποιεί κάθε χρονοδρομολογητής (βραχυχρόνιος, μεσοχρόνιος ή μακροχρόνιος) βασίζεται στον *αλγόριθμο χρονοδρομολόγησης* (scheduling algorithm) τον οποίο εφαρμόζει.

Ο αλγόριθμος αυτός περιγράφει με σαφήνεια τον τρόπο με τον οποίο ο χρονοδρομολογητής θα επιλέξει μια διεργασία για να την προωθήσει στην ΚΜΕ ή να την απομακρύνει από αυτή, πόσο χρόνο θα απασχολεί κάθε διεργασία την ΚΜΕ κλπ. Οι παράμετροι που λαμβάνει υπόψη του ο αλγόριθμος είναι το πλήθος των διεργασιών, η διάρκεια που αναμένεται να έχουν οι εκρήξεις ΚΜΕ και Ε/Ε για τις διεργασίες αυτές, κλπ.

Οι βασικοί στόχοι ενός αλγορίθμου χρονοδρομολόγησης είναι δυο:

⇨ Το υπολογιστικό σύστημα πρέπει να έχει όσο το δυνατόν καλύτερη απόδοση σύμφωνα με κάποια κριτήρια αξιολόγησης

⇨ Ο ίδιος ο αλγόριθμος πρέπει να λαμβάνει τις αποφάσεις του όσο πιο γρήγορα γίνεται, ιδιαίτερα όταν το πλήθος των διεργασιών είναι μεγάλο, για να προσθέτει την ελάχιστη επιβάρυνση στο σύστημα

Ο δεύτερος στόχος είναι ιδιαίτερα σημαντικός για το βραχυχρόνιο χρονοδρομολογητή, ο οποίος εκτελείται πολύ συχνά.

Κριτήρια αξιολόγησης για αλγορίθμους χρονοδρομολόγησης

Τα ιδιαίτερα χαρακτηριστικά των αλγορίθμων χρονοδρομολόγησης μπορεί να ευνοούν κάποια ομάδα διεργασιών έναντι άλλων. Για να αξιολογηθεί λοιπόν η επίδοσή τους, χρειάζονται κάποια «αντικειμενικά» κριτήρια, τα οποία μετρούν τις επιδόσεις του υπολογιστικού συστήματος. Υπάρχουν διάφορα τέτοια κριτήρια, αλλά συνήθως δεν είναι δυνατόν ένας αλγόριθμος να είναι αποδοτικός σε όλες τις περιπτώσεις. Έτσι, ανάλογα με το σκοπό για τον οποίο θα χρησιμοποιηθεί ένα υπολογιστικό σύστημα, επιλέγονται τα κριτήρια που ταιριάζουν στο σκοπό αυτό και ο αλγόριθμος χρονοδρομολόγησης που τα ικανοποιεί.

Τα πιο συνηθισμένα κριτήρια αξιολόγησης είναι:

1. Ο **βαθμός χρησιμοποίησης της ΚΜΕ** (CPU utilization).
   Πρόκειται για το ποσοστό του χρόνου που η ΚΜΕ είναι απασχολημένη. Αν το υπολογιστικό σύστημα λειτουργεί για χρονικό διάστημα $T_{ολ}$ και η ΚΜΕ έχει απασχοληθεί για χρόνο $T_{απ}$, ο βαθμός χρησιμοποίησής της είναι $T_{απ}$ / $T_{ολ}$. Οι καλοί βαθμοί χρησιμοποίησης είναι κοντά στο 1 (όπου η ΚΜΕ είναι συνεχώς απασχολημένη).
2. Η **ρυθμαπόδοση** (throughput).
   Είναι το πλήθος των εργασιών που ολοκληρώνονται στη μονάδα του χρόνου. Αν π.χ. μέσα σε μια ώρα ολοκληρωθούν 1800 εργασίες, τότε η ρυθμαπόδοση του συστήματος είναι 0,5 εργασίες/sec. Όσο πιο μεγάλη είναι η ρυθμαπόδοση, τόσο καλύτερη γίνεται η απόδοση του συστήματος.
3. Ο **χρόνος ανακύκλωσης** (turnaround time).
   Είναι ο συνολικός χρόνος που απαιτείται για την πλήρη εκτέλεση μιας εργασίας, από τη στιγμή που αυτή υποβάλλεται στο σύστημα μέχρι τη στιγμή που ολοκληρώνεται. Αυτός ο χρόνος περιλαμβάνει την αρχική αναμονή της εργασίας μέχρι να επιλεγεί για φόρτωση στην κύρια μνήμη, το χρόνο εκτέλεσης στην ΚΜΕ, το χρόνο Ε/Ε και το χρόνο αναμονής σε διάφορες ουρές του συστήματος.
4. Ο **χρόνος αναμονής** (waiting time).
   Πρόκειται για το χρόνο που πέρασε η διεργασία στη λίστα έτοιμων διεργασιών αναμένοντας την εκτέλεσή της. Ο χρόνος αναμονής είναι ένας «καλύτερος» δείκτης της καθυστέρησης που επιβάλλει το σύστημα σε μια διεργασία από το χρόνο ανακύκλωσης, γιατί ο δεύτερος περιέχει και το χρόνο που εκτελείται η διεργασία στην ΚΜΕ ή εκτελεί λειτουργίες Ε/Ε, ο οποίος είναι σταθερός για κάθε διεργασία και δεν εξαρτάται από το ΛΣ.
5. Ο **χρόνος απόκρισης** (response time).
   Σε διαλογικά συστήματα ενδιαφέρει πολλές φορές ο χρόνος που απαιτείται μέχρι το σύστημα να δώσει την πρώτη έξοδο-απόκριση για μια διεργασία, που ονομάζεται χρόνος απόκρισης. Αυτός διαφέρει από το χρόνο ανακύκλωσης κατά το χρονικό διάστημα που απαιτείται για την ολοκλήρωση της εξόδου μιας διεργασίας, και συνεπώς δεν εξαρτάται από την ταχύτητα της συσκευής εξόδου. Αν π.χ. μια διεργασία μετά το τέλος των υπολογισμών της εκτυπώνει τα αποτελέσματα, ο χρόνος απόκρισης μετριέται μέχρι τη στιγμή που αρχίζει η εκτύπωση· ο χρόνος της εκτύπωσης δεν συνυπολογίζεται.

Κατηγορίες αλγορίθμων χρονοδρομολόγησης

Οι αλγόριθμοι χρονοδρομολόγησης μπορούν να ταξινομηθούν σε δυο μεγάλες κατηγορίες:

- ***Μη διακοπτοί (non preemptive) αλγόριθμοι***: Μια διεργασία που επιλέγεται να πάρει τον έλεγχο της ΚΜΕ, τον διατηρεί έως ότου αυτή ολοκληρωθεί, ή απαιτήσει κάποια άλλη λειτουργία (π.χ. Ε/Ε).
- ***Διακοπτοί (preemptive) αλγόριθμοι***: Αφού μια διεργασία χρησιμοποιήσει την ΚΜΕ εκτελούμενη για κάποιο χρονικό διάστημα (το κβάντο χρόνου), το ΛΣ τη διακόπτει, ανεξάρτητα από το αν έχει ολοκληρωθεί, για να δώσει την ΚΜΕ σε κάποια άλλη.

Σε ένα μη διακοπτό αλγόριθμο δηλαδή, κάθε διεργασία εκτελεί χωρίς διακοπή ολόκληρες τις εκρήξεις ΚΜΕ, ενώ σε ένα διακοπτό τις εκτελεί τμηματικά.

Μη διακοπτοί αλγόριθμοι χρονοδρομολόγησης

Οι πιο διαδεδομένοι μη διακοπτοί αλγόριθμοι χρονοδρομολόγησης είναι τρεις, και βασίζονται, αντίστοιχα, στη σειρά άφιξης των διεργασιών, τη διάρκειά τους και το λόγο απόκρισής τους. Θα εφαρμόσουμε κάθε έναν από τους αλγορίθμους στο ίδιο παράδειγμα για να συγκρίνουμε τις επιδόσεις τους. Ας υποθέσουμε λοιπόν ότι οι τέσσερις διεργασίες προς εξυπηρέτηση είναι οι $δ_1$, $δ_2$, $δ_3$ και $δ_4$. Η $δ_1$ φθάνει στη λίστα έτοιμων διεργασιών τη χρονική στιγμή 0 και έχει διάρκεια 12 χρονικές μονάδες, η $δ_2$ φθάνει τη στιγμή 6 και έχει διάρκεια 31, η $δ_3$ φθάνει τη στιγμή 8 και έχει διάρκεια 3 και η $δ_4$ φθάνει τη στιγμή 14 και έχει διάρκεια 11. Επιπλέον γίνεται η παραδοχή ότι οι διεργασίες απασχολούν μόνο την ΚΜΕ και δεν εκτελούν λειτουργίες Ε/Ε.

Διεργασία	Άφιξη	Διάρκεια
$δ_1$	0	12
$δ_2$	6	31
$δ_3$	8	3
$δ_4$	14	11

Εξυπηρέτηση με βάση τη σειρά άφιξης

> Ο πιο απλός τρόπος επιλογής του επόμενου πελάτη του ζαχαροπλαστείου που θα εξυπηρετηθεί είναι να επιλέγεται αυτός που περιμένει περισσότερη ώρα. Όση ώρα το γλυκό του παρασκευάζεται οι υπόλοιποι περιμένουν τη σειρά τους. Για να βρίσκεται εύκολα ο επόμενος πελάτης που θα εξυπηρετηθεί, όποιος μπαίνει στο ζαχαροπλαστείο περιμένει στην ουρά για να έρθει η σειρά του.

Ο αντίστοιχος αλγόριθμος χρονοδρομολόγησης ΚΜΕ ονομάζεται *αλγόριθμος με βάση τη σειρά άφιξης* (First Come First Served - FCFS). Η υλοποίηση του γίνεται πολύ εύκολα· η λίστα έτοιμων διεργασιών λειτουργεί όπως μια ουρά αναμονής.

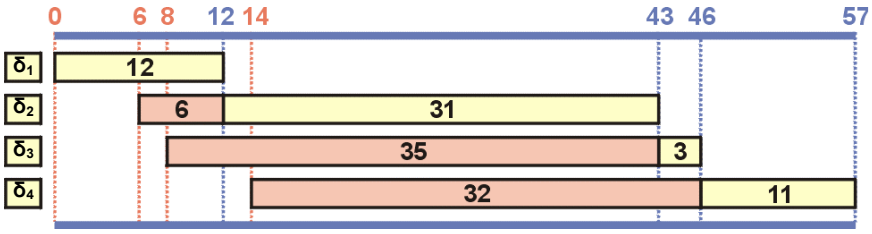
Κάθε νέα διεργασία που ετοιμάζεται να εκτελέσει την επόμενη έκρηξή της στην ΚΜΕ εισάγεται στο τέλος της λίστας έτοιμων διεργασιών. Όταν έρθει η ώρα να εισαχθεί στην ΚΜΕ μια νέα διεργασία, επιλέγεται αυτή που είναι στην αρχή τής λίστας, προωθείται στην ΚΜΕ και εκεί εκτελεί ολόκληρη την έκρηξή της στην ΚΜΕ.

Η επίδοση του αλγορίθμου αυτού αν κριθεί με βάση το *χρόνο ανακύκλωσης* (turnaround time) είναι χαμηλή, γιατί εξαναγκάζει ακόμα και διεργασίες με πολύ σύντομες εκρήξεις ΚΜΕ να αναμένουν για την εκτέλεση άλλων. Παρόμοιες επιδόσεις πετυχαίνει και με βάση τα υπόλοιπα κριτήρια αξιολόγησης που αναφέρθηκαν στο μάθημα αυτό· η χρησιμοποίηση π.χ. της ΚΜΕ μπορεί να είναι χαμηλή.

Το βασικό πλεονέκτημα της εξυπηρέτησης με βάση τη σειρά άφιξης είναι η απλότητά της. Ο αλγόριθμος είναι πάρα πολύ απλός να προγραμματιστεί και καταναλώνει ελάχιστο χρόνο για την εκτέλεσή του.

Στο σχήμα φαίνεται το σχεδιάγραμμα εκτέλεσης των τεσσάρων διεργασιών του παραδείγματος. Οι χρόνοι αναμονής είναι 0 για την πρώτη διεργασία, 12-6=6 για τη δεύτερη, 12+31-8=35 για την τρίτη και 12+31+3-14=32 για την τέταρτη. Ο μέσος

χρόνος αναμονής είναι λοιπόν (0+6+35+32)/4 = 18,25. Παρατηρούμε επίσης ότι η διεργασία δ_3, η οποία είχε διάρκεια μόνο 3, εξαναγκάστηκε να περιμένει για 35 χρονικές μονάδες για να εκτελεστεί, είχε δηλαδή τεράστια αναμονή σε σχέση με τη διάρκειά της.



Οι χρόνοι απόκρισης (χρόνος αναμονής + χρόνος εκτέλεσης) είναι: 12 για τη δ_1, 37 για τη δ_2, 38 για τη δ_3 και 43 για τη δ_4. Έτσι ο μέσος χρόνος απόκρισης είναι (12+37+38+43)/4 = 32,5.

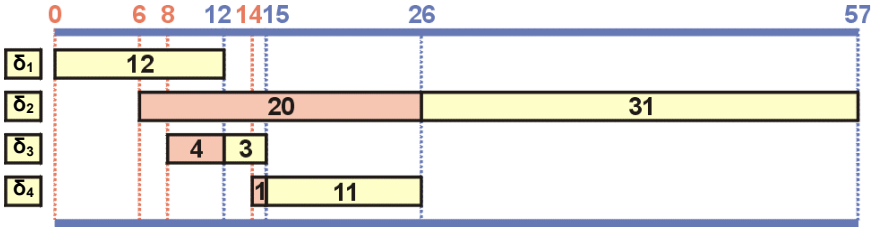
Εξυπηρέτηση με βάση τη διάρκεια

> Καλό είναι οι πελάτες που επιθυμούν την παρασκευή κάποιων απλών γλυκών που απαιτούν λίγο χρόνο, να μην περιμένουν πολλή ώρα μέχρι να εκτελεστούν κάποιες άλλες, χρονοβόρες, συνταγές. Αντί αυτού, είναι προτιμότερο να περιμένουν πολλή ώρα αυτοί που έχουν παραγγείλει μια χρονοβόρα συνταγή. Γενικά, όταν προηγούνται αυτοί που θα εξυπηρετηθούν σύντομα, κανείς δεν περιμένει άδικα για ένα υπερβολικό χρονικό διάστημα*.

Ο αλγόριθμος εξυπηρέτησης *με βάση τη διάρκεια* (Shortest Job First - SJF), κατατάσσει τις διεργασίες κατά αύξουσα σειρά διάρκειας. Πρώτα εκτελείται η συντομότερη διεργασία, μετά αυτή με την αμέσως μεγαλύτερη διάρκεια κ.ο.κ. Τελευταία εκτελείται αυτή με τη μεγαλυτερη διάρκεια.

Μια διεργασία που εισάγεται στη λίστα έτοιμων διεργασιών, όταν μια άλλη ήδη εκτελείται, θα τοποθετηθεί στο κατάλληλο σημείο ανάλογα με τη διάρκεια της έκρηξής της στην ΚΜΕ, και θα εκτελεστεί πριν από κάποια άλλη που είχε εισαχθεί στη λίστα νωρίτερα αλλά είχε μεγαλύτερη διάρκεια.

Όσον αφορά το χρόνο που οι διεργασίες βρίσκονται σε αναμονή, ο αλγόριθμος με βάση τη διάρκεια είναι βέλτιστος, δηλαδή κανένας άλλος αλγόριθμος δεν μπορεί να επιτύχει καλύτερο μέσο χρόνο αναμονής.



Η εκτέλεση των διεργασιών του παραδείγματος με τον αλγόριθμο εξυπηρέτησης με βάση τη διάρκεια φαίνεται στο σχήμα. Η σειρά εκτέλεσης των διεργασιών είναι δ_1, δ_3, δ_4, δ_2. Η δ_1 επειδή είναι μόνη της πρώτη δεν περιμένει καθόλου, η δ_2 περιμένει για (12+3+11)-6 = 20, η δ_3 για 12-8 = 4 και η δ_4 για (12+3)-14 = 1. Ο μέσος χρόνος αναμονής είναι λοιπόν (0+20+4+1)/4 = 6,25· πολύ μικρότερος από τον αντίστοιχο για την εξυπηρέτηση με βάση τη σειρά άφιξης, και ο μικρότερος δυνατός.

Οι χρόνοι απόκρισης είναι: 12 για τη δ_1, 51 για τη δ_2, 7 για τη δ_3 και 12 για τη δ_4. Ο μέσος χρόνος απόκρισης είναι (12+51+7+12)/4 = 20,5, πολύ καλύτερος επίσης από την εξυπηρέτηση με βάση τη σειρά άφιξης.

---

* Η αρχή αυτή εφαρμόζεται σε ορισμένα μεγάλα καταστήματα όπου οι πελάτες που αγοράζουν μικρό αριθμό αντικειμένων εξυπηρετούνται σε ιδιαίτερα ταμεία.

Η δυσκολία στην υλοποίηση του αλγορίθμου αυτού είναι η εκτίμηση της διάρκειας για την έκρηξη ΚΜΕ κάθε διεργασίας. Στην περίπτωση του μακροχρόνιου χρονοδρομολογητή, οι χρήστες μπορούν να έχουν θέσει εκ των προτέρων ανώτατα όρια στο χρόνο εκτέλεσης του προγράμματός τους, με βάση τα οποία να γίνεται η επιλογή της συντομότερης διεργασίας. Όμως στο βραχυχρόνιο χρονοδρομολογητή δεν είναι εύκολο να υπολογιστεί εκ των προτέρων η επόμενη έκρηξη ΚΜΕ για μια διεργασία· μπορεί βέβαια να εκτιμηθεί στατιστικά με βάση τις προηγούμενες εκρήξεις ΚΜΕ, αλλά η εκτίμηση αυτή δεν είναι ακριβής και επιπλέον απαιτεί υπολογιστικό χρόνο.

Εξυπηρέτηση με βάση το λόγο απόκρισης

Στους δυο αλγορίθμους που παρουσιάστηκαν οι παράμετροι που λαμβάνονται υπόψη είναι δυο:

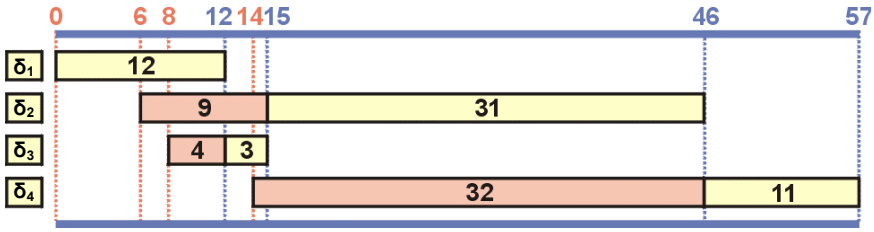
- Στην εξυπηρέτηση με βάση τη σειρά άφιξης, ο αλγόριθμος εξετάζει τον πραγματικό χρόνο αναμονής για κάθε μια διεργασία και επιλέγει αυτή με το μεγαλύτερο χρόνο αναμονής.
- Στην εξυπηρέτηση με βάση τη διάρκεια, ο αλγόριθμος ελέγχει τη διάρκεια της επόμενης έκρηξης ΚΜΕ και επιλέγει τη διεργασία με τη μικρότερη έκρηξη.

Βλέπουμε λοιπόν ότι οι δυο αλγόριθμοι είναι μάλλον μονόπλευροι ως προς τη χρονοδρομολόγηση ΚΜΕ. Μια μέση λύση θα προσπαθούσε να λάβει υπόψη και τις δυο αυτές παραμέτρους, συνδυάζοντάς τις σε μια. Αυτή η παράμετρος είναι ο *λόγος απόκρισης* (response ratio) και είναι:

$$\textit{Λόγος Απόκρισης} = \frac{\textit{Χρόνος Απόκρισης}}{\textit{Χρόνος Εκτέλεσης}} = \frac{\textit{Χρόνος Εκτέλεσης} + \textit{Χρόνος Αναμονής}}{\textit{Χρόνος Εκτέλεσης}} = 1 + \frac{\textit{Χρόνος Αναμονής}}{\textit{Χρόνος Εκτέλεσης}}$$

Ο αλγόριθμος που βασίζεται στο λόγο απόκρισης ονομάζεται αλγόριθμος *εξυπηρέτησης με βάση το λόγο απόκρισης* (Highest Response Ratio First, HRF), και επιλέγει για εκτέλεση εκείνη τη διεργασία που έχει το μεγαλύτερο λόγο απόκρισης.

Στο παράδειγμά μας, η διεργασία δ_1 που φθάνει πρώτη για εκτέλεση καταλαμβάνει την ΚΜΕ. Όσο αυτή εκτελείται, φθάνουν οι δ_2 και δ_3. Όταν η δ_1 ολοκληρωθεί, η δ_2 θα έχει χρόνο αναμονής 6 και η δ_3 θα έχει 4. Ο λόγος απόκρισης για τη δ_2 θα είναι λοιπόν $1+6/31 \cong 1{,}19$ και για τη δ_3 θα είναι $1+4/3 \cong 2{,}33$. Επιλέγεται λοιπόν η δ_3 για εκτέλεση, κατά τη διάρκεια της οποίας φθάνει η δ_4. Όταν και η δ_3 τελειώσει τη χρονική στιγμή 15, η δ_2 θα έχει περάσει 15-6 = 9 χρονικές μονάδες σε αναμονή και η δ_4 1. Ο λόγος απόκρισης για τη δ_2 θα είναι τώρα $1+9/31 \cong 1{,}29$ και για τη δ_4 θα είναι $1+1/11 \cong 1{,}09$. Αυτή τη φορά λοιπόν επιλέγεται η δ_2 για εκτέλεση και τελευταία εκτελείται η δ_4. Η ακολουθία εκτέλεσης φαίνεται στο σχήμα.



Ο χρόνος αναμονής για τη δ_2 είναι (12+3)-6 = 9, για τη δ_3 είναι 12-8 = 4 και για τη δ_4 είναι (12+3+31)-14 = 32. Έτσι ο μέσος χρόνος αναμονής είναι (0+9+4+32)/4 = 11,25· μεγαλύτερος από το χρόνο της

εξυπηρέτησης με βάση τη διάρκεια, αλλά καλύτερος από την εξυπηρέτηση με βάση τη σειρά άφιξης.

Οι χρόνοι απόκρισης εδώ είναι 12 για τη δ_1, 40 για τη δ_2, 7 για τη δ_3 και 43 για τη δ_4. Έτσι ο μέσος χρόνος απόκρισης γίνεται (12+40+7+43)/4 = 25,5.

Συνοπτικά, τα αποτελέσματα για τους τρεις αλγορίθμους φαίνονται στον πίνακα που ακολουθεί. Καλύτερος από τους τρεις για το συγκεκριμένο παράδειγμα είναι ο αλγόριθμος εξυπηρέτησης με βάση τη διάρκεια και για τα δυο κριτήρια αξιολόγησης.

	Με βάση τη σειρά άφιξης (FCFS)	Με βάση τη διάρκεια (SJF)	Με βάση το λόγο απόκρισης (HRF)
Μέσος χρόνος αναμονής	18,25	6,25	11,25
Μέσος χρόνος απόκρισης	32,50	20,50	24,50

Εξυπηρέτηση με βάση την προτεραιότητα

Οι τρεις μη διακοπτοί αλγόριθμοι που περιγράφηκαν στις προηγούμενες παραγράφους αποτελούν ειδική περίπτωση ενός γενικότερου αλγόριθμου, ο οποίος δρομολογεί διεργασίες με βάση την προτεραιότητά (priority) τους.

Στον αλγόριθμο αυτό, σε κάθε διεργασία αντιστοιχίζεται μια αριθμητική τιμή, που είναι η προτεραιότητά της, και η διεργασία με τη μεγαλύτερη προτεραιότητα επιλέγεται κάθε φορά για εκτέλεση. Όταν δυο ή περισσότερες διεργασίες έχουν την ίδια προτεραιότητα, τότε επιλέγεται πρώτα η παλαιότερη, δηλαδή χρονοδρομολογούνται με βάση τη σειρά άφιξης.

Ας δούμε πώς εκφράζονται οι τρεις προηγούμενοι αλγόριθμοι με βάση την προτεραιότητα:

- ⇨ Στην εξυπηρέτηση με βάση τη σειρά άφιξης, όλες οι διεργασίες έχουν την ίδια προτεραιότητα, π.χ. 1. Έτσι κάθε φορά επιλέγεται για εκτέλεση η παλαιότερη.
- ⇨ Στην εξυπηρέτηση με βάση τη διάρκεια, θέλουμε να εκτελείται πρώτη, να έχει δηλαδή τη *μεγαλύτερη* προτεραιότητα, η διεργασία με τη *μικρότερη* έκρηξη ΚΜΕ. Πρέπει δηλαδή η προτεραιότητα να είναι μια ποσότητα που μεγαλώνει όταν η έκρηξη ΚΜΕ μικραίνει. Η ποσότητα 1/(έκρηξη ΚΜΕ) έχει αυτή την ιδιότητα, αφού είναι αντιστρόφως ανάλογη της έκρηξης ΚΜΕ και επιλέγεται για να παραστήσει την προτεραιότητα.
- ⇨ Στην εξυπηρέτηση με βάση το λόγο απόκρισης, θέλουμε να έχει τη μεγαλύτερη προτεραιότητα η διεργασία με το μεγαλύτερο λόγο απόκρισης. Έτσι πολύ απλά μπορούμε να επιλέξουμε το λόγο απόκρισης για να παίξει το ρόλο της προτεραιότητας.

Στους δυο πρώτους αλγορίθμους, η προτεραιότητα μιας διεργασίας καθορίζεται μια φορά, κατά την είσοδό της στη λίστα έτοιμων διεργασιών και δεν αλλάζει ξανά. Η προτεραιότητα που δεν αλλάζει κατά τη διάρκεια της ζωής μιας διεργασίας ονομάζεται *στατική προτεραιότητα* (static priority). Αντίθετα, στον αλγόριθμο εξυπηρέτησης με βάση το λόγο απόκρισης, οι προτεραιότητες (δηλαδή οι λόγοι απόκρισης) των διεργασιών υπολογίζονται κάθε φορά που πρέπει να επιλεγεί μια διεργασία για εκτέλεση. Η προτεραιότητα που αλλάζει κατά τη ζωή μιας διεργασίας καλείται *δυναμική προτεραιότητα* (dynamic priority).

Ένα πολύ ουσιαστικό πρόβλημα σε ορισμένους αλγορίθμους χρονοδρομολόγησης είναι ότι κάποια διεργασία με αρκετά μικρή προτεραιότητα μπορεί να αναμένει για την εκτέλεσή της επ' αόριστο. Κυρίως όταν οι προτεραιότητες είναι στατικές (όπως στην

εξυπηρέτηση με βάση τη διάρκεια), σε ένα υπερφορτωμένο σύστημα, μια διεργασία με χαμηλή προτεραιότητα μπορεί πρακτικά να είναι αποκλεισμένη από την ΚΜΕ γιατί συνεχώς την υπερπηδούν άλλες, υψηλότερης προτεραιότητας.

Μια λύση που μπορεί να δοθεί στο πρόβλημα αυτό είναι η τεχνική της *γήρανσης* (aging), η οποία αυξάνει την προτεραιότητα μιας διεργασίας με την πάροδο του χρόνου, ώστε κάποια στιγμή να εκτελεστεί. Αυτή την τεχνική εφαρμόζει ουσιαστικά η εξυπηρέτηση με βάση το λόγο απόκρισης: όσο περνά ο χρόνος και μια διεργασία δεν εκτελείται, ο λόγος απόκρισής της (άρα και η προτεραιότητά της) αυξάνει.

Διακοπτοί αλγόριθμοι χρονοδρομολόγησης

Στους διακοπτούς αλγορίθμους μια διεργασία δεν καταλαμβάνει την ΚΜΕ για όλη τη διάρκεια της έκρηξης ΚΜΕ. Το ΛΣ μπορεί να τη διακόψει και να παραχωρήσει την ΚΜΕ σε κάποια άλλη διεργασία, η οποία (όπως κρίνει το ΛΣ) πρέπει να προηγηθεί στην εκτέλεσή της.

> Την ώρα που ο ζαχαροπλάστης (ΚΜΕ) εκτελεί ένα στάδιο μιας συνταγής (έκρηξη ΚΜΕ), μπαίνει στο κατάστημα ένας πελάτης που βιάζεται πολύ. Οι οδηγίες (ΛΣ) προς το ζαχαροπλάστη για την εξυπηρέτηση των πελατών προβλέπουν ότι κάθε φορά που ένας νέος πελάτης έρχεται, εξετάζεται η συνταγή και οι ανάγκες του. Αν ικανοποιεί κάποιες προϋποθέσεις, τον εξυπηρετεί αμέσως, διακόπτοντας αυτόν που εξυπηρετούσε μέχρι τότε. Οι προϋποθέσεις που πρέπει να ικανοποιεί ο πελάτης για να του παραχωρηθεί η προτεραιότητα αποτελούν το συγκεκριμένο διακοπτό αλγόριθμο χρονοδρομολόγησης.

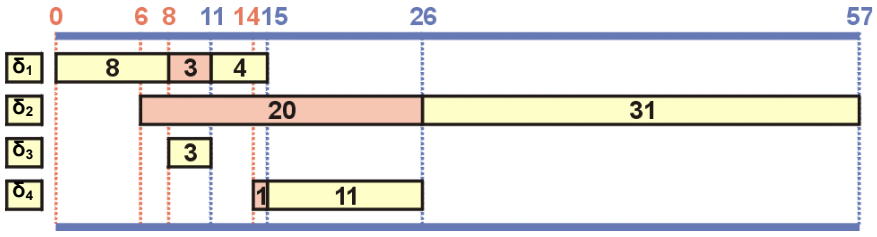
Διακοπτή εξυπηρέτηση με βάση τη διάρκεια

> Κάθε φορά που ένας νέος πελάτης μπαίνει στο ζαχαροπλαστείο, ο ζαχαροπλάστης υπολογίζει το χρόνο που χρειάζεται για να τον εξυπηρετήσει, και το συγκρίνει με το χρόνο που χρειάζεται για να ολοκληρώσει τη συνταγή την οποία εκτελεί εκείνη τη στιγμή. Αν ο νέος πελάτης απαιτεί λιγότερο χρόνο, τότε σταματά τη συνταγή που εκτελούσε και ξεκινά τη δική του.

Στο διακοπτό αλγόριθμο εξυπηρέτησης *με βάση τη διάρκεια* (Preemptive Shortest Job First), κάθε φορά που μια νέα διεργασία εισέρχεται στη λίστα έτοιμων διεργασιών για εκτέλεση, ελέγχεται αν η διάρκεια της έκρηξης ΚΜΕ που θα εκτελέσει είναι μικρότερη από το χρόνο που απομένει στη διεργασία η οποία εκτελείται. Στην περίπτωση αυτή, η διεργασία που εκτελείται διακόπτεται αμέσως και η νέα διεργασία παίρνει τη θέση της. Η διεργασία που διακόπηκε τοποθετείται πάλι στη λίστα έτοιμων διεργασιών.

Όταν μια διεργασία ολοκληρώσει την εκτέλεσή της, επιλέγεται από τη λίστα έτοιμων διεργασιών εκείνη με τη συντομότερη έκρηξη ΚΜΕ και της παραχωρείται η ΚΜΕ, όπως ακριβώς και στη μη διακοπτή παραλλαγή του αλγορίθμου.

Στο παράδειγμα που χρησιμοποιήθηκε και προηγουμένως, η διεργασία $δ_1$ φθάνει τη χρονική στιγμή 0 και αρχίζει να εκτελείται. Τη χρονική στιγμή 6 φθάνει η διεργασία $δ_2$, που θα έχει διάρκεια 31 μονάδες. Στη $δ_1$ απομένουν 6 μονάδες για να τελειώσει, λιγότερες από το χρόνο της $δ_2$, οπότε η $δ_1$ συνεχίζει να



εκτελείται. Τη χρονική στιγμή 8 όμως φθάνει η διεργασία δ_3 με διάρκεια 3, ενώ στην δ_1 απομένουν 4 χρονικές μονάδες εκτέλεσης. Έτσι η δ_1 διακόπτεται και αρχίζει να εκτελείται η δ_3, η οποία ολοκληρώνεται τη στιγμή 11. Τότε υπάρχουν δυο διεργασίες για εκτέλεση: η δ_1, στην οποία απομένουν 4 χρονικές μονάδες για να ολοκληρωθεί, και η δ_2 η οποία απαιτεί 31 χρονικές μονάδες. Επιλέγεται βέβαια η δ_1 και αρχίζει να εκτελείται. Τη χρονική στιγμή 14, και ενώ η δ_1 εκτελείται ακόμα, φθάνει στη λίστα έτοιμων διεργασιών η δ_4 που απαιτεί 11 χρονικές μονάδες. Αφού στη δ_1 απομένει μόνο μια χρονική μονάδα το ΛΣ δεν τη διακόπτει και την αφήνει να ολοκληρωθεί. Όταν τελειώσει, οι υποψήφιες διεργασίες είναι: η δ_2 με διάρκεια 31 και η δ_4 με διάρκεια 11. Επιλέγεται η δ_4 και αφού ολοκληρωθεί εκτελείται και η δ_2.

Η δ_1 περνά σε αναμονή για 3 χρονικές μονάδες, κατά τις οποίες εκτελείται η δ_3 (που δεν περιμένει καθόλου). Η δ_2 περιμένει για 20 χρονικές μονάδες και η δ_4 για μια. Ο μέσος χρόνος αναμονής είναι (3+20+0+1)/4 = 6. Αυτός είναι καλύτερος και από τον καλύτερο μη διακοπτό αλγόριθμο, τον SJF, που είχε μέσο χρόνο αναμονής 6,25.

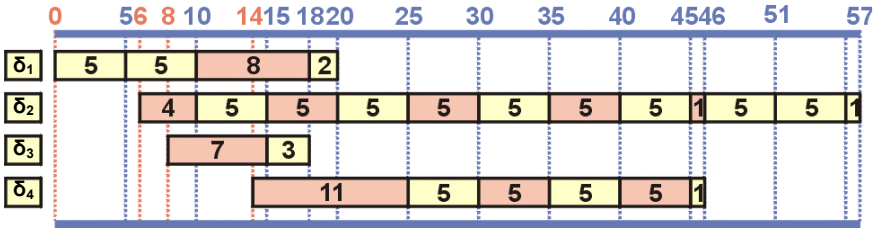
Η διεργασία δ_1 έχει χρόνο απόκρισης 15, η δ_2 51, η δ_3 3 και η δ_4 12. Έτσι, ο μέσος χρόνος απόκρισης είναι (15+51+3+12)/4 = 20,25, καλύτερος και αυτός από όλους τους μη διακοπτούς αλγορίθμους (αν και η διαφορά του από το μη διακοπτό SJF είναι ελάχιστη). Βέβαια στους υπολογισμούς μας δεν έχουμε λάβει υπόψη την επιβάρυνση που προκαλεί η εκτέλεση του χρονοδρομολογητή.

<u>Χρονοδρομολόγηση κυκλικής επαναφοράς</u>

Ο *αλγόριθμος κυκλικής επαναφοράς* (Round Robin - RR) έχει σχεδιασθεί ειδικά για συστήματα καταμερισμού χρόνου (time sharing) όπου δίνεται ένα μικρό ποσό χρόνου εναλλάξ σε κάθε διεργασία. Αυτό το μικρό ποσό χρόνου, όπως ήδη έχουμε πει, ονομάζεται *κβάντο χρόνου* (time quantum) και έχει συνήθως διάρκεια της τάξης των 10 msec. Η λίστα έτοιμων διεργασιών είναι η ουρά στην οποία περιμένουν οι διεργασίες. Κάθε φορά που περνά ένα κβάντο χρόνου, η ΚΜΕ διακόπτει την τρέχουσα διεργασία και την τοποθετεί στο τέλος της λίστας έτοιμων διεργασιών. Αφαιρεί στη συνέχεια τη διεργασία που βρίσκεται στην αρχή της ουράς και την παραχωρεί στην ΚΜΕ για το επόμενο κβάντο χρόνου.

Αν μια διεργασία ολοκληρώσει την εκτέλεσή της προτού τελειώσει το κβάντο χρόνου που της έχει παραχωρηθεί, απελευθερώνει οικειοθελώς την ΚΜΕ και το ΛΣ παραχωρεί την ΚΜΕ στην επόμενη έτοιμη διεργασία.

Αν υπάρχει μόνο μια διεργασία στο σύστημα, τότε της παραχωρούνται διαδοχικά κβάντα χρόνου.



Το πλάνο εκτέλεσης του παραδείγματος με τον αλγόριθμο κυκλικής επαναφοράς φαίνεται στο σχήμα, για κβάντο 5 μονάδων χρόνου. Παρατηρούμε ότι οι διεργασίες που εισάγονται στη λίστα έτοιμων διεργασιών την ώρα που κάποια άλλη εκτελείται πρέπει να περιμένουν να έρθει η σειρά τους για εκτέλεση μετά από 1, 2 ή και 3 κβάντα χρόνου. Αυτό συμβαίνει γιατί κάθε νέα διεργασία τοποθετείται στο τέλος της λίστας έτοιμων διεργασιών, οπότε πρέπει πρώτα να εκτελεστούν όλες όσες προηγούνται (για ένα κβάντο χρόνου η κάθε μια) και μετά αυτή.

Επίσης βλέπουμε ότι, αν μια διεργασία ολοκληρωθεί πριν τελειώσει το κβάντο χρόνου της (π.χ. η δ_3 τελειώνει τη χρονική στιγμή 18 ενώ το κβάντο χρόνου τελείωνε στην 20),

η ΚΜΕ παραχωρείται αμέσως για ένα ολόκληρο κβάντο χρόνου στην επόμενη διεργασία.

Όπως φαίνεται στο σχήμα, η δ_1 περνά συνολικά 8 χρονικές μονάδες σε αναμονή, η δ_2 περνά 20 χρονικές μονάδες, η δ_3 7 και η δ_4 20. Ο μέσος χρόνος αναμονής είναι λοιπόν (3+20+7+20)/4 = 12,5 μονάδες χρόνου. Αυτός είναι ένας μεγάλος μέσος χρόνος αναμονής, αλλά αντισταθμίζεται από το γεγονός ότι στους χρήστες δίνεται η εντύπωση ότι οι διεργασίες τους εκτελούνται όντως παράλληλα.

Από το σχήμα επίσης μπορούμε να υπολογίσουμε τους χρόνους απόκρισης για τις τέσσερις διεργασίες: 20 για τη δ_1, 51 για τη δ_2, 10 για τη δ_3 και 31 για τη δ_4. Ο μέσος χρόνος απόκρισης είναι λοιπόν (20+51+10+31)/4 = 28.

Η επίδοση του αλγορίθμου κυκλικής επαναφοράς επηρεάζεται σημαντικά από την επιλογή της διάρκειας που θα έχει το κβάντο χρόνου.

Αν αυτό έχει πολύ μεγάλη τιμή, τότε ο αλγόριθμος πρακτικά εκφυλίζεται στον αλγόριθμο εξυπηρέτησης με βάση τη σειρά άφιξης (FCFS), ο οποίος όπως είδαμε δεν έχει πολύ καλές επιδόσεις. Αν όμως, από την άλλη, το κβάντο χρόνου είναι πολύ μικρό, γίνονται πολλές εναλλαγές από τη μια διεργασία στην άλλη, οι οποίες επιβαρύνουν αρκετά το σύστημα. Πρέπει λοιπόν η τιμή του κβάντου χρόνου να είναι τέτοια ώστε η επιβάρυνση της εναλλαγής μεταξύ των διεργασιών να μην είναι σπουδαία. Ένας καλός κανόνας για τον προσδιορισμό ενός ικανοποιητικού κβάντου χρόνου απαιτεί να είναι μεγαλύτερο σε διάρκεια από το 80% των εκρήξεων ΚΜΕ των διαφόρων διεργασιών.

Το κυριότερο πλεονέκτημα του αλγορίθμου κυκλικής επαναφοράς είναι ότι διευκολύνει τις μικρές διεργασίες να εκτελεστούν, διότι αυτές ολοκληρώνονται σχετικά γρήγορα, χωρίς να περάσουν πολύ χρόνο σε αναμονή.

Ανακεφαλαίωση

Η μεθοδολογία που ακολουθεί το ΛΣ για να κάνει τη χρονοδρομολόγηση, δηλαδή ο αλγόριθμος χρονοδρομολόγησης που χρησιμοποιεί, μπορεί να αξιολογηθεί με διάφορα κριτήρια: το χρόνο απόκρισης, ανακύκλωσης ή αναμονής, τη ρυθμαπόδοση ή τη χρησιμοποίηση της ΚΜΕ. Οι αλγόριθμοι, ανάλογα με το αν επιτρέπουν στις διεργασίες να εκτελέσουν ακέραιες τις εκρήξεις ΚΜΕ ή όχι διακρίνονται σε μη διακοπτούς και διακοπτούς αντίστοιχα. Οι τρεις κυριότεροι μη διακοπτοί αλγόριθμοι χρονοδρομολόγησης επιλέγουν τη διεργασία που θα εκτελεστεί στην ΚΜΕ (για όλη την έκρηξη ΚΜΕ αφού είναι μη διακοπτοί) με βάση τη σειρά άφιξης των διεργασιών στη λίστα έτοιμων διεργασιών, τη διάρκεια της έκρηξης ΚΜΕ ή το λόγο απόκρισης. Και οι τρεις μπορούν να θεωρηθούν εφαρμογές της χρονοδρομολόγησης με βάση την προτεραιότητα. Οι διακοπτοί αλγόριθμοι χρονοδρομολόγησης επιτρέπουν τη διακοπή μιας διεργασίας προτού αυτή ολοκληρώσει την έκρηξη ΚΜΕ της, για να παραχωρηθεί η ΚΜΕ σε κάποια άλλη διεργασία. Οι πιο διαδεδομένοι διακοπτοί αλγόριθμοι είναι ο διακοπτός SJF (με βάση τη διάρκεια) και ο αλγόριθμος κυκλικής επαναφοράς που χρησιμοποιείται στα συστήματα καταμερισμού χρόνου.

Γλωσσάριο όρων

Αλγόριθμος Κυκλικής Επαναφοράς	*Round Robin - RR*
Αλγόριθμος Χρονοδρομολόγησης	*Scheduling Algorithm*
Βαθμός Χρησιμοποίησης της ΚΜΕ	*CPU Utilization*
Γήρανση	*Aging*
Διακοπτή Εξυπηρέτηση με Βάση τη Διάρκεια	*Preemptive Shortest Job First*
Διακοπτός Αλγόριθμος	*Preemptive Algorithm*
Δυναμική Προτεραιότητα	*Dynamic Priority*
Εξυπηρέτηση με Βάση τη Διάρκεια	*Shortest Job First - SJF*
Εξυπηρέτηση με Βάση τη Σειρά Άφιξης	*First Come First Served - FCFS*
Εξυπηρέτηση με Βάση το Λόγο Απόκρισης	*Highest Response Ratio First - HRF*
Λόγος Απόκρισης	*Response Ratio*
Μη Διακοπτός Αλγόριθμος	*Non Preemptive Algorithm*
Προτεραιότητα	*Priority*
Ρυθμαπόδοση	*Throughput*
Στατική Προτεραιότητα	*Static Priority*
Χρόνος Ανακύκλωσης	*Turnaround Time*
Χρόνος Αναμονής	*Waiting Time*
Χρόνος Απόκρισης	*Response Time*

Ερωτήσεις

? Ποια είναι τα βασικά κριτήρια αξιολόγησης για αλγορίθμους χρονοδρομολόγησης;

? Ποιες είναι οι βασικές κατηγορίες αλγορίθμων χρονοδρομολόγησης;

? Ανάφερε τους αλγορίθμους χρονοδρομολόγησης που υπάρχουν σε μη διακοπτή και σε διακοπτή έκδοση.

? Ανάφερε τους αλγορίθμους χρονοδρομολόγησης που υπάρχουν μόνο σε διακοπτή έκδοση.

Τι μάθαμε σε αυτό το κεφάλαιο

- Η χρονοδρομολόγηση σε ένα ΛΣ αποφασίζει ποια διεργασία θα χρησιμοποιεί ανά πάσα στιγμή την ΚΜΕ.
- Η χρονοδρομολόγηση μπορεί να γίνει στο βραχυχρόνιο, το μακροχρόνιο και το μεσοχρόνιο επίπεδο.
- Οι αλγόριθμοι χρονοδρομολόγησης διακρίνονται σε διακοπτούς και μη διακοπτούς.
- Μερικοί αλγόριθμοι χρονοδρομολόγησης είναι η εξυπηρέτηση με βάση τη διάρκεια, το λόγο απόκρισης, την άφιξη, την προτεραιότητα και η κυκλική εναλλαγή.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Ν. Μπιλάλη, Π. Τσανάκα, Λειτουργικά Συστήματα: Αρχές Λειτουργίας, Συμμετρία, 1989.
- Silberschatz A., Peterson J., Galvin P., Operating System Concepts, Addison - Wesley, 1991.

Κεφάλαιο

Διαχείριση Μνήμης στα Λειτουργικά Συστήματα

Σκοπός του κεφαλαίου αυτού είναι να σου γνωρίσει τον τρόπο με τον οποίο ένα λειτουργικό σύστημα διαχειρίζεται τη μνήμη, ώστε οι διεργασίες να χρησιμοποιούν τη μνήμη με απλό τρόπο, και η απόδοση του συστήματος να είναι βέλτιστη.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να περιγράφεις τον τρόπο που το ΛΣ εκμεταλλεύεται τη δευτερεύουσα μνήμη για να ενισχύσει το μέγεθος της κύριας.
- Να εξηγείς τι είναι εικονική μνήμη και πώς αντιστοιχίζεται στη φυσική μνήμη.
- Να απαριθμείς τα προβλήματα που καλείται να λύσει η διαχείριση μνήμης του ΛΣ και να περιγράφεις τις λύσεις που δίνονται στα προβλήματα αυτά.

Μαθήματα

9.1 Διαχείριση Μνήμης

9.2 Τεχνικές Διαχείρισης Εικονικής Μνήμης

Μάθημα 9.1 Διαχείριση Μνήμης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να εξηγήσει τον τρόπο που ένα ΛΣ χειρίζεται τις ανάγκες των διεργασιών για μνήμη.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να απαριθμείς τα είδη μνήμης σε ένα υπολογιστή*
- *Να εξηγείς πώς το ΛΣ εκμεταλλεύεται τη δευτερεύουσα μνήμη για να ενισχύσει το μέγεθος της κύριας*
- *Να ορίζεις την εικονική μνήμη*
- *Να περιγράφεις πώς διαχειριζόμαστε την εικονική μνήμη και πώς την αντιστοιχίζουμε στη φυσική μνήμη*

Η μνήμη ενός ανθρώπου τον βοηθά να συγκρατεί όλες τις χρήσιμες πληροφορίες που έχει συγκεντρώσει με τη βοήθεια των αισθήσεών του από τον κόσμο που τον περιβάλλει. Αντίστοιχα ένας υπολογιστής πρέπει να διαθέτει έναν τρόπο να καταγράφει διάφορα στοιχεία: είτε τα προγράμματα που εκτελεί, είτε τα δεδομένα των προγραμμάτων αυτών.

Κάθε μονάδα ενός υπολογιστή που χρησιμεύει για τη μόνιμη ή προσωρινή αποθήκευση δεδομένων ανήκει στη *μνήμη* (memory) του υπολογιστή.

Οι μονάδες μνήμης ανήκουν σε δυο αρκετά διαφορετικές κατηγορίες ανάλογα με το αν αποθηκεύουν τα δεδομένα προσωρινά ή μόνιμα:

➲ Στην *κύρια* ή *κεντρική μνήμη* (main memory) του υπολογιστή αποθηκεύονται τα προγράμματα που εκτελεί η ΚΜΕ και τα δεδομένα για τα προγράμματα αυτά. Η αποθήκευση αυτή είναι προσωρινή, και διαρκεί όσο και η λειτουργία του υπολογιστή. Όταν κλείσει ο υπολογιστής τα δεδομένα αυτά χάνονται. Επειδή η κύρια μνήμη των υπολογιστών είναι έτσι οργανωμένη ώστε να μπορεί να προσπελαστεί άμεσα οποιαδήποτε θέση της, αναφέρεται ως *μνήμη τυχαίας προσπέλασης* (Random Access Memory) και αποκαλείται RAM[1].

Συμπλήρωμα της κύριας μνήμης είναι η *μνήμη ROM* (Read Only Memory), στην οποία είναι καταγραμμένα μόνιμα από τον κατασκευαστή του υπολογιστή ορισμένα βασικά προγράμματα ή μικρά τμήματα του ΛΣ.

[1] Οι όροι «κύρια μνήμη» και «RAM» δεν είναι εξ ορισμού ταυτόσημοι, αλλά έχει καθιερωθεί η από κοινού χρήση τους για την περιγραφή της κύριας μνήμης.

➲ Στη *δευτερεύουσα* ή *περιφερειακή μνήμη* (secondary memory) τα δεδομένα αποθηκεύονται μόνιμα. Τα δεδομένα που θα καταγραφούν στη δευτερεύουσα μνήμη δε χάνονται όταν κλείσει ο υπολογιστής, και είναι διαθέσιμα την επόμενη φορά που θα λειτουργήσει. Μονάδες δευτερεύουσας μνήμης είναι οι σκληροί δίσκοι, οι εύκαμπτοι δίσκοι, τα CD ROM κλπ.

Η φυσική μνήμη του ζαχαροπλάστη είναι η κύρια μνήμη του, ενώ δευτερεύουσα μνήμη για αυτόν είναι ένα σημειωματάριο με αριθμημένες σελίδες που περιέχει τις συνταγές του ζαχαροπλάστη και λευκές σελίδες για τις σημειώσεις του. Όταν ο ζαχαροπλάστης διαβάζει μια συνταγή και τη θυμάται για λίγη ώρα, την μεταφέρει στην κύρια μνήμη του από τη δευτερεύουσα. Επίσης θυμάται διάφορες πληροφορίες για την πρόοδο των συνταγών. Επειδή δεν μπορεί να θυμάται όλες τις συνταγές που εκτελεί ταυτόχρονα και όλες τις σχετικές πληροφορίες, όταν πρόκειται να διαβάσει μια νέα συνταγή καταγράφει στο σημειωματάριο τις πληροφορίες που χρειάζεται για κάποιες άλλες (όπως π.χ. σε ποιο σημείο της εκτέλεσής τους βρίσκονται) ώστε να μην τις ξεχάσει.

Όταν ψάχνει κάποια πληροφορία στο σημειωματάριό του, ο ζαχαροπλάστης δε διαβάζει μία σελίδα για να διαπιστώσει αν είναι αυτή που θέλει, αλλά τη βρίσκει απευθείας από τον αριθμό της σελίδας της, το όνομα του πελάτη ή το όνομα της συνταγής.

Στην πράξη έχει καθιερωθεί η χρήση του όρου μνήμη για την κύρια μνήμη, ενώ οι μονάδες δευτερεύουσας μνήμης αναφέρονται με το όνομά τους (π.χ. δίσκος). Έτσι, στη συνέχεια όταν αναφερόμαστε στη μνήμη του υπολογιστή θα εννοούμε την κύρια μνήμη.

Στη συντριπτική πλειοψηφία των σύγχρονων υπολογιστών, τα προγράμματα που εκτελεί ο υπολογιστής και τα δεδομένα που χρειάζονται ή δημιουργούν τα προγράμματα αυτά είναι καταχωρημένα στη μνήμη με τον ίδιο ακριβώς τρόπο. Το ΛΣ κρατά πληροφορίες για τη μνήμη, ώστε να μπορεί να διακρίνει σε ποιες θέσεις της βρίσκονται προγράμματα και σε ποιες βρίσκονται δεδομένα.

Για να εκτελεστεί μια διεργασία πρέπει τόσο το πρόγραμμά της όσο και τα δεδομένα της να έχουν καταχωρηθεί σε κάποιες θέσεις της κύριας μνήμης. Για να εκτελεστεί η διεργασία, είναι απαραίτητο να χρησιμοποιηθούν το πρόγραμμα και τα δεδομένα. Οι αντίστοιχες θέσεις μνήμης διαβάζονται και γράφονται, γίνονται λοιπόν *προσπελάσεις* (accesses) σε αυτές.

Είναι φανερό ότι ο τρόπος με τον οποίο το ΛΣ διαχειρίζεται την κύρια μνήμη ενός υπολογιστή επηρεάζει πολύ τον τρόπο λειτουργίας του, αλλά και την απόδοσή του. Όταν σχεδιάζεται ένα ΛΣ, η διαχείριση της κύριας μνήμης είναι ένα πεδίο σοβαρών σχεδιαστικών αποφάσεων:

☒ Κάθε στιγμή θα χρησιμοποιούν τη μνήμη μια ή περισσότερες διεργασίες;

☒ Αν η μνήμη χρησιμοποιείται από πολλές διεργασίες ταυτόχρονα, θα είναι χωρισμένη σε τμήματα ίσου ή διαφορετικού μεγέθους;

☒ Ο χωρισμός της μνήμης σε τμήματα θα είναι μόνιμος ή θα αλλάζει ανάλογα με τις διεργασίες που εκτελούνται;

☒ Το πρόγραμμα και τα δεδομένα κάθε διεργασίας θα είναι αποθηκευμένα σε ένα συνεχόμενο τμήμα της μνήμης ή θα μπορούν να κατανεμηθούν ανάμεσα σε πολλά διάσπαρτα τμήματα;

Στατική και δυναμική κατανομή της κύριας μνήμης

Για να μπορούν πολλές διεργασίες να εκτελούνται ταυτόχρονα σε ένα υπολογιστή, πρέπει η κύρια μνήμη του να είναι διαιρεμένη σε τμήματα και σε καθένα από αυτά να βρίσκεται το πρόγραμμα ή τα δεδομένα μίας διεργασίας. Με τον τρόπο αυτό, η ΚΜΕ μπορεί να εναλλάσσεται μεταξύ των διεργασιών, διαβάζοντας από και γράφοντας σε διαφορετικές θέσεις της μνήμης κάθε φορά.

Επίσης, για να λειτουργεί ο υπολογιστής ομαλά, κάθε διεργασία πρέπει να έχει πρόσβαση μόνο σε εκείνα τα τμήματα της μνήμης που της αντιστοιχούν και όχι στα τμήματα άλλων διεργασιών. Ιδιαίτερα προστατευμένα πρέπει να είναι τα τμήματα της μνήμης όπου είναι αποθηκευμένος ο κώδικας και τα δεδομένα του ίδιου του ΛΣ.

Η κατανομή της κύριας μνήμης μπορεί να είναι *στατική* (static allocation) ή *δυναμική* (dynamic allocation).

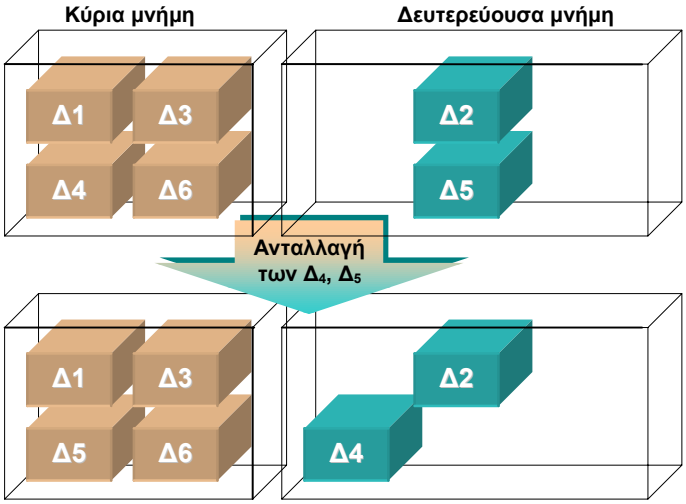
Στη **στατική κατανομή** η μνήμη είναι χωρισμένη από το ΛΣ σε προκαθορισμένα τμήματα πριν από την εκτέλεση των διεργασιών. Όταν ένα πρόγραμμα πρόκειται να εκτελεστεί, το ΛΣ επιλέγει ένα τμήμα μνήμης που καλύπτει τις ανάγκες της νέας διεργασίας σε μνήμη και της το παραχωρεί. Η στατική κατανομή μνήμης είναι απλή και λύνει το πρόβλημα της προστασίας της μνήμης. Όμως, τα τμήματα μνήμης δε χρησιμοποιούνται πλήρως από τις διεργασίες και έτσι μένουν αναξιοποίητα. Επιπλέον οι διεργασίες δεν μπορούν να χρησιμοποιήσουν μνήμη από κοινού, κάτι που είναι χρήσιμο όταν πολλές διεργασίες εκτελούν το ίδιο πρόγραμμα.

Στη **δυναμική κατανομή** το ΛΣ κάνει τη διαίρεση της μνήμης κάθε φορά που μία διεργασία αρχίζει να εκτελείται και της διαθέτει ακριβώς όση μνήμη χρειάζεται. Η δυναμική κατανομή της μνήμης είναι ευέλικτη και μπορεί να αξιοποιήσει μικρές περιοχές της μνήμης. Με τον τρόπο αυτό βελτιώνεται το πρόβλημα της αναξιοποίητης μνήμης, χωρίς όμως να λύνεται τελείως. Το βασικό μειονέκτημα της λύσης αυτής είναι η αυξημένη πολυπλοκότητα των μηχανισμών προστασίας της μνήμης.

Ανταλλαγή

Η δευτερεύουσα μνήμη είναι πολύ πιο αργή από την κύρια (π.χ. ένας σκληρός δίσκος είναι εκατό χιλιάδες φορές πιο αργός από την κύρια μνήμη)· συγχρόνως όμως είναι και αρκετά πιο φθηνή. Έτσι, οι περισσότεροι υπολογιστές διαθέτουν πολύ περισσότερο χώρο αποθήκευσης σε δευτερεύουσα μνήμη παρά σε κύρια (π.χ. εκατό φορές περισσότερο).

Όταν ένας υπολογιστής εξυπηρετεί ένα μεγάλο αριθμό διεργασιών, τα προγράμματα και τα δεδομένα όλων δε χωρούν στην κύρια μνήμη. Τότε επιστρατεύεται η δευτερεύουσα μνήμη για να βοηθήσει: τα προγράμματα και τα δεδομένα ορισμένων διεργασιών κρατούνται στη δευτερεύουσα μνήμη, και κάθε φορά που είναι η σειρά μιας τέτοιας διεργασίας να εκτελεστεί, τότε μόνο φορτώνονται στην κύρια μνήμη. Για να απελευθερωθεί όμως χώρος στην κύρια μνήμη για αυτά, πρέπει κάποια άλλη διεργασία να μεταφερθεί με τη σειρά της στη δευτερεύουσα μνήμη.

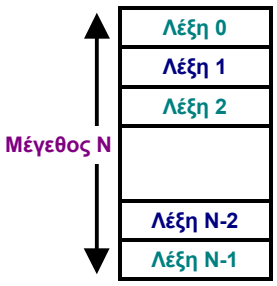


Η διαδικασία αυτή ονομάζεται *ανταλλαγή* (swapping) και δημιουργεί σοβαρές χρονικές καθυστερήσεις στην εκτέλεση των διεργασιών, εξαιτίας της συνεχούς λειτουργίας της δευτερεύουσας μνήμης. Ο χρόνος που απαιτεί μια διεργασία

για να εκτελεστεί είναι το άθροισμα του πραγματικού χρόνου εκτέλεσής της στην ΚΜΕ, του χρόνου που απαιτήθηκε για τις μεταφορές της μεταξύ κύριας και βοηθητικής μνήμης και του χρόνου για Ε/Ε. Για να είναι η λειτουργία του υπολογιστή αποδοτική απαιτείται πιο πολύπλοκη οργάνωση της ανταλλαγής δεδομένων μεταξύ κύριας και δευτερεύουσας μνήμης.

Εικονική μνήμη

Η κεντρική μνήμη του υπολογιστή αποτελείται από διαδοχικές θέσεις, με μέγεθος η κάθε μια π.χ. 16 ή 32 bits, που ονομάζονται *λέξεις* (words). Κάθε λέξη της μνήμης έχει τη δική της διεύθυνση, με την οποία αναφέρονται τα προγράμματα σε αυτή. Αν η μνήμη έχει μέγεθος N λέξεων, τότε η πρώτη λέξη έχει διεύθυνση 0, η δεύτερη έχει διεύθυνση 1 κ.ο.κ., ενώ η τελευταία λέξη έχει διεύθυνση N-1. Το σύνολο αυτών των N διευθύνσεων είναι σταθερό και ονομάζεται *χώρος φυσικών διευθύνσεων* (physical address space) ή *χώρος απολύτων διευθύνσεων* (absolute address space) ή *χώρος πραγματικών διευθύνσεων* (real address space).



Κάθε πρόγραμμα πρέπει να χρησιμοποιήσει διευθύνσεις για να αναφερθεί στα δεδομένα του, τα οποία βρίσκονται στη μνήμη. Τι γίνεται όμως αν οι διευθύνσεις των δεδομένων για δυο διεργασίες συμπίπτουν;

> Μια διεργασία $δ_1$ υπολογίζει ένα άθροισμα, το οποίο τοποθετεί στη θέση μνήμης με διεύθυνση 156. Μια άλλη διεργασία $δ_2$ ζητά διάφορους αριθμούς από το χρήστη, τους οποίους τοποθετεί επίσης στη θέση μνήμης με διεύθυνση 156. Οι δυο διευθύνσεις συμπίπτουν, γιατί τα δυο προγράμματα γράφτηκαν ανεξάρτητα το ένα από το άλλο. Αν και οι δυο διεργασίες εκτελούνται ταυτόχρονα, η θέση μνήμης 156 θα χρησιμοποιείται πότε από τη μια διεργασία και πότε από την άλλη. Έτσι όμως καμία από τις δυο δε θα χρησιμοποιεί τις σωστές τιμές, αφού σε τυχαίες χρονικές στιγμές επεμβαίνει η άλλη και τις αλλάζει, και τελικά καμία δε θα δώσει τα σωστά αποτελέσματα.

Τι γίνεται επίσης όταν η διαθέσιμη κύρια μνήμη δε χωρά όλες τις διεργασίες;

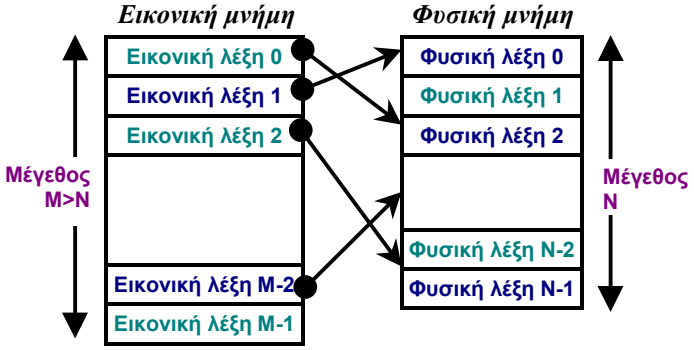
Εικονικές διευθύνσεις

Η πρώτη και πιο απλή σκέψη που μπορεί να γίνει για να λυθεί το πρόβλημα της επικάλυψης των διευθύνσεων είναι να γράφονται έτσι τα προγράμματα ώστε καθένα να χρησιμοποιεί διαφορετικές διευθύνσεις μνήμης. Με τον τρόπο αυτό δεν υπάρχει περίπτωση η μια διεργασία να επηρεάσει την άλλη. Αυτή η λύση όμως δεν είναι στην πράξη και πολύ αποτελεσματική για δυο λόγους:

- Τα προγράμματα γράφονται συνήθως ανεξάρτητα από διάφορους ανθρώπους, και δεν είναι δυνατό να βρεθεί ένας τρόπος συνεννόησης και συντονισμού τους.
- Το πλήθος των διαθέσιμων διευθύνσεων κύριας μνήμης είναι πεπερασμένο και μάλλον μικρό, οπότε αρκετά γρήγορα θα εξαντληθούν.

Θέλουμε λοιπόν μια λύση που να επιτρέπει στα προγράμματα να χρησιμοποιούν ελεύθερα οποιεσδήποτε διευθύνσεις, και να εξασφαλίζεται από το ΛΣ ότι δε θα υπάρχει επικάλυψη μεταξύ τους. Η λύση αυτή είναι εκείνη των *εικονικών διευθύνσεων* (virtual addresses): κάθε πρόγραμμα μεταφράζεται σε γλώσσα μηχανής σαν να έχει όλο το χώρο διευθύνσεων στη διάθεσή του, και αναφέρεται στα δεδομένα του χρησιμοποιώντας εικονικές -δηλαδή όχι πραγματικές- διευθύνσεις· όταν το πρόγραμμα φορτώνεται για εκτέλεση, το ΛΣ επιλέγει θέσεις μνήμης που είναι ελεύθερες και τις αντιστοιχίζει στις εικονικές διευθύνσεις.

Κατά την εκτέλεση του προγράμματος, κάθε αναφορά σε μια εικονική διεύθυνση μεταφράζεται από το ΛΣ στην αντίστοιχη φυσική διεύθυνση, η οποία τελικά προσπελάζεται. Η διαδικασία αυτή της μετάφρασης εικονικής διεύθυνσης σε φυσική συνήθως γίνεται πολύ γρήγορα, γιατί οι ΚΜΕ περιέχουν ειδικά κυκλώματα για αυτή.



Ο *χώρος εικονικών διευθύνσεων* (virtual address space), που πολλές φορές αναφέρεται και ως *εικονική μνήμη* (virtual memory), έχει συνήθως μεγαλύτερο μέγεθος από το χώρο φυσικών διευθύνσεων για να έχουν οι διεργασίες περισσότερο «χώρο μνήμης» στη διάθεσή τους. Οι εικονικές διευθύνσεις που δεν έχουν αντίστοιχες στη φυσική μνήμη, συνήθως αντιστοιχίζονται σε κάποια διεύθυνση της δευτερεύουσας μνήμης. Με τον τρόπο αυτό η δευτερεύουσα μνήμη συμβάλλει στην αύξηση της διαθέσιμης κύριας μνήμης του συστήματος.

Ο τρόπος αντιστοίχισης των διευθύνσεων για διαφορετικά προγράμματα μπορεί να γίνει με διάφορους τρόπους, δηλαδή με διάφορες στρατηγικές διαχείρισης εικονικής μνήμης. Οι βασικές στρατηγικές είναι δυο, η *σελιδοποίηση* (paging) και η *κατάτμηση* (segmentation), και υπάρχει ένας συνδυασμός των δύο, η *κατατμημένη σελιδοποίηση* (segmented paging). Αυτές οι στρατηγικές έχουν την ιδιότητα να επιτρέπουν πολλές διεργασίες να χρησιμοποιούν από κοινού μνήμη, όπως π.χ. διεργασίες που εκτελούν ταυτόχρονα το ίδιο πρόγραμμα χρησιμοποιούν από κοινού τη μνήμη στην οποία είναι αυτό αποθηκευμένο.

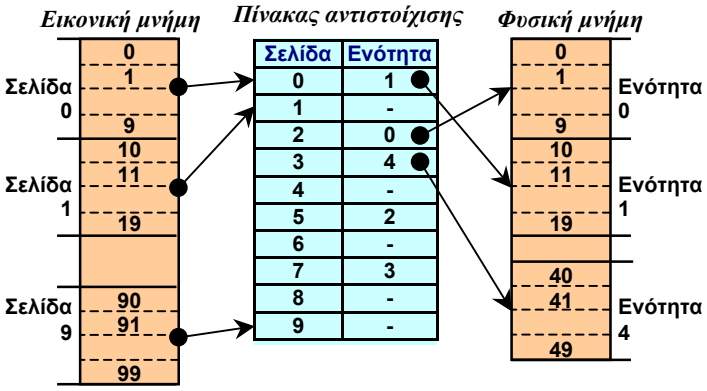
Διαχείριση εικονικής μνήμης με σελιδοποίηση

Στη μέθοδο της σελιδοποίησης, η εικονική μνήμη διαιρείται σε ίσα και συνεχόμενα μέρη, τα οποία ονομάζονται *σελίδες* (pages). Με τον ίδιο τρόπο διαιρείται και η φυσική μνήμη σε *ενότητες* (blocks, frames). Το μέγεθος της ενότητας είναι ίδιο με αυτό της σελίδας· έτσι μια σελίδα της εικονικής μνήμης και όλες οι διευθύνσεις που αυτή περιέχει αντιστοιχούν ακριβώς σε μια ενότητα της φυσικής μνήμης και τις διευθύνσεις της.

Το ΛΣ κρατά ένα πίνακα αντιστοίχισης σελίδων και ενοτήτων. Για κάθε σελίδα της εικονικής μνήμης φαίνεται στον πίνακα αυτό η ενότητα της φυσικής μνήμης στην οποία αντιστοιχεί ή, αν δεν αντιστοιχεί σε κάποια ενότητα, σημειώνεται ότι η σελίδα βρίσκεται στη δευτερεύουσα μνήμη.

Για να μεταφραστεί μια εικονική διεύθυνση (ΕΔ), πρώτα βρίσκεται η σελίδα (Σ) στην οποία ανήκει. Το ΛΣ αναζητά τη σελίδα αυτή στον πίνακα αντιστοίχισης, και βρίσκει ότι αντιστοιχεί στην ενότητα Ε. Η φυσική διεύθυνση (ΦΔ) θα ανήκει στην ενότητα αυτή, και θα βρίσκεται σε αντίστοιχη θέση με αυτή της ΕΔ μέσα στη σελίδα Σ.

Σε έναν υπολογιστή η εικονική μνήμη έχει μέγεθος 100 (εικονικές διευθύνσεις από 0 ως 99) και χωρίζεται σε 10 σελίδες ενώ η φυσική μνήμη έχει μέγεθος 50 (φυσικές διευθύνσεις από 0 ως 49) και διαιρείται σε 5 ενότητες. Το μέγεθος σελίδων και ενοτήτων είναι 10. Έτσι π.χ. οι εικονικές διευθύνσεις από 0 ως 9 ανήκουν στη σελίδα με αριθμό 0, ενώ οι φυσικές διευθύνσεις από 30 ως 39



ανήκουν στην ενότητα με αριθμό 3.

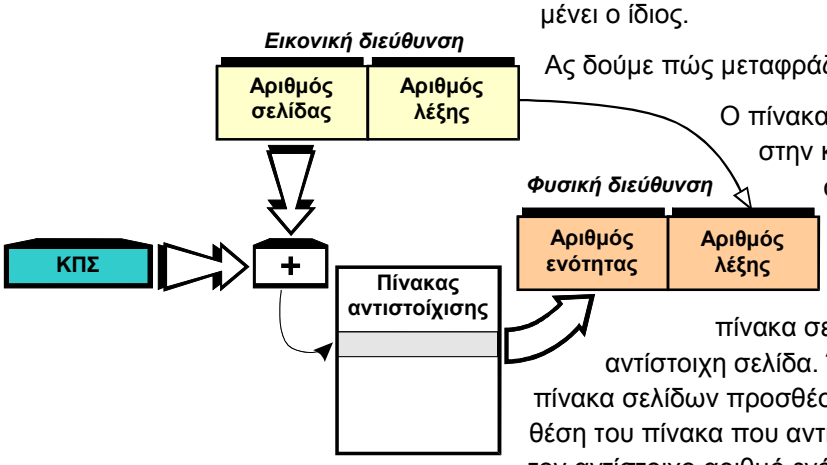
Στον πίνακα αντιστοίχισης του σχήματος, η σελίδα 0 αντιστοιχεί στην ενότητα 1, δηλαδή οι εικονικές διευθύνσεις 0-9 αντιστοιχούν στις φυσικές διευθύνσεις 10-19. Η εικονική διεύθυνση 2 τελικά θα μεταφραστεί στη φυσική διεύθυνση 12.

Παρατηρούμε ότι κάθε διεύθυνση μπορεί να χωριστεί σε δυο τμήματα: οι δεκάδες της διεύθυνσης δείχνουν τη σελίδα ή την ενότητα στην οποία ανήκει η διεύθυνση, και οι μονάδες δείχνουν τη θέση της διεύθυνσης μέσα στη σελίδα ή την ενότητα. Έτσι για να υπολογίσουμε τη φυσική διεύθυνση ΦΔ από την εικονική διεύθυνση ΕΔ, ακολουθούμε τα εξής βήματα:

1. Βρίσκουμε τον αριθμό σελίδας Σ, που είναι οι δεκάδες της ΕΔ.
2. Βρίσκουμε τη θέση Θ της διεύθυνσης μέσα στη σελίδα, που είναι οι μονάδες της ΕΔ.
3. Από τον πίνακα αντιστοίχισης, βρίσκουμε τον αριθμό ενότητας Ε για τη φυσική μνήμη.
4. Η ΦΔ έχει για δεκάδες τον αριθμό ενότητας Ε και για μονάδες τη θέση Θ μέσα στην ενότητα.

Για την εικονική διεύθυνση ΕΔ = 5, ο αριθμός Σ της σελίδας είναι 0 και η θέση Θ είναι 5. Από τον πίνακα βρίσκουμε ότι η αντίστοιχη ενότητα Ε είναι η 1, έτσι η φυσική διεύθυνση θα είναι 15.

Κάθε εικονική διεύθυνση σε έναν υπολογιστή χωρίζεται σε δύο τμήματα: το πρώτο είναι ο αριθμός της σελίδας και το δεύτερο ο αριθμός της λέξης μέσα στη σελίδα. Στο παράδειγμά μας, κάθε τέτοιο τμήμα ήταν ένα δεκαδικό ψηφίο· στους υπολογιστές κάθε τμήμα της εικονικής διεύθυνσης αποτελείται από πολλά ψηφία, αφενός γιατί χρησιμοποιείται το δυαδικό σύστημα, αφετέρου γιατί το μέγεθος της εικονικής μνήμης είναι μεγάλο. Αντίστοιχα και κάθε φυσική διεύθυνση χωρίζεται στον αριθμό ενότητας και τον αριθμό λέξης. Όταν μεταφράζεται μια εικονική διεύθυνση σε φυσική, ο αριθμός λέξης μένει ο ίδιος.



Ας δούμε πώς μεταφράζεται μια εικονική διεύθυνση σε φυσική:

Ο πίνακας αντιστοίχισης της διεργασίας βρίσκεται στην κύρια μνήμη, και η διεύθυνση όπου αρχίζει κρατείται σε μια μονάδα αποθήκευσης της ΚΜΕ που ονομάζεται *καταχωρητής πίνακα σελίδων* (ΚΠΣ, page table register). Κάθε «θέση» του πίνακα σελίδων κρατά τον αριθμό ενότητας για την αντίστοιχη σελίδα. Έτσι, αν στην αρχική διεύθυνση του πίνακα σελίδων προσθέσουμε ένα αριθμό σελίδας, θα πάρουμε τη θέση του πίνακα που αντιστοιχεί στη σελίδα αυτή, και θα βρούμε τον αντίστοιχο αριθμό ενότητας για τη σελίδα.

Αν ο πίνακας σελίδων αρχίζει στη διεύθυνση 10 της μνήμης, ο αριθμός ενότητας για τη σελίδα 0 βρίσκεται στη διεύθυνση 10, ο αριθμός ενότητας για τη σελίδα 1 βρίσκεται στη διεύθυνση 11, κλπ. Γενικά ο αριθμός ενότητας για τη σελίδα κ βρίσκεται στη διεύθυνση 10+κ.

Για να βρούμε λοιπόν τον αριθμό ενότητας για μια εικονική διεύθυνση προσθέτουμε τον αριθμό σελίδας της με το περιεχόμενο του καταχωρητή πίνακα σελίδων. Το αποτέλεσμα είναι μια διεύθυνση μνήμης, και στη διεύθυνση αυτή θα βρούμε τον αριθμό ενότητας στη φυσική μνήμη. Αυτός ο αριθμός ενότητας συνδυάζεται με τον αριθμό λέξης για να δώσουν την τελική φυσική διεύθυνση.

Αν για μια σελίδα δεν υπάρχει αντίστοιχη ενότητα στην κύρια μνήμη, τότε πρέπει να ενεργοποιηθεί μια διαδικασία μεταφοράς της ενότητας αυτής από τη δευτερεύουσα στην κύρια μνήμη, ενώ συγχρόνως κάποια άλλη σελίδα μεταφέρεται από την κύρια στη δευτερεύουσα μνήμη για να απελευθερώσει την ενότητα που χρειάζεται. Η θέση στη δευτερεύουσα μνήμη όπου βρίσκονται όσες σελίδες είναι εκτός κύριας μνήμης μπορεί να καταγράφεται στον πίνακα σελίδων ή σε κάποιον άλλο πίνακα.

Ο πίνακας σελίδων είναι και αυτός αποθηκευμένος στη μνήμη. Έτσι κάθε μετάφραση εικονικής διεύθυνσης σε φυσική απαιτεί μια επιπλέον προσπέλαση μνήμης για την ανάγνωση του πίνακα[2]. Αυτή η προσπέλαση εισάγει μια ανεπιθύμητη καθυστέρηση στην εκτέλεση των εντολών. Μια λύση στο πρόβλημα αυτό είναι η χρήση ειδικής, γρήγορης μνήμης, για την αποθήκευση του πίνακα σελίδων.

Το κυριότερο μειονέκτημα της σελιδοποίησης είναι ότι κάθε διεργασία καταλαμβάνει περισσότερο χώρο από ό,τι χρειάζεται. Συνήθως η μνήμη που απαιτεί μια διεργασία δεν είναι πολλαπλάσιο του μεγέθους της σελίδας, αλλά χρειάζεται π.χ. 10 σελίδες ολόκληρες και 20 λέξεις επιπλέον. Επειδή ο χώρος εικονικών διευθύνσεων που της δίνεται έχει μέγεθος που είναι πολλαπλάσιο της σελίδας, θα της δοθούν 11 σελίδες· από την 11η θα χρησιμοποιηθούν μόνο οι 20 πρώτες λέξεις. Η τελευταία σελίδα κάθε διεργασίας έτσι έχει ένα τμήμα, μικρότερο ή μεγαλύτερο, το οποίο μένει αχρησιμοποίητο. Αυτό το φαινόμενο ονομάζεται *εσωτερικός κατακερματισμός* (internal fragmentation).

Διαχείριση εικονικής μνήμης με κατάτμηση και κατατμημένη σελιδοποίηση

Η σελιδοποίηση είναι η πιο κοινά χρησιμοποιούμενη μέθοδος για τη διαχείριση της εικονικής μνήμης, γιατί συνήθως υποστηρίζεται από το υλικό των υπολογιστών. Δυο άλλες μέθοδοι που χρησιμοποιούνται, αλλά όχι τόσο ευρέως, είναι η κατάτμηση και η κατατμημένη σελιδοποίηση· η δεύτερη είναι συνδυασμός κατάτμησης και σελιδοποίησης.

Η μέθοδος της κατάτμησης προσπαθεί να αποφύγει τον εσωτερικό κατακερματισμό δίνοντας σε κάθε διεργασία ακριβώς όση μνήμη της χρειάζεται. Χωρίζει τη μνήμη σε τμήματα διαφορετικών μεγεθών, και για κάθε ένα από αυτά κρατά την αρχική διεύθυνση και το μέγεθός του. Για να μεταφράσει μια εικονική διεύθυνση σε φυσική, αναζητεί τις πληροφορίες που έχει κρατήσει για το αντίστοιχο τμήμα, ελέγχει αν η δεδομένη διεύθυνση είναι μέσα στα όρια του τμήματος, βρίσκει την αρχική του διεύθυνση στη φυσική μνήμη, και τέλος τη συνδυάζει με τη θέση μνήμης μέσα στο τμήμα για να υπολογίσει την τελική φυσική διεύθυνση.

Όταν δημιουργείται μια νέα διεργασία, το ΛΣ αναζητά ένα τμήμα εικονικών διευθύνσεων αρκετά μεγάλο για να χωρέσει τη μνήμη που χρειάζεται η διεργασία, και της αποδίδει τις διευθύνσεις αυτές.

---

[2] Εδώ έχουμε κάνει την απλουστευτική παραδοχή ότι μία γραμμή του πίνακα αντιστοίχισης χωρά ακριβώς σε μία λέξη της μνήμης. Αν μία γραμμή του πίνακα είναι μεγάλη και απαιτεί περισσότερες από μία λέξεις, τότε χρειάζονται οι αντίστοιχες επιπλέον προσπελάσεις στη μνήμη.

Η κατατμημένη σελιδοποίηση συνδυάζει τις δυο τεχνικές, εργαζόμενη πάλι με τμήματα, τα οποία όμως αποτελούνται από σελίδες σταθερού μεγέθους. Η κατάτμηση έτσι είναι η ειδική περίπτωση της κατατμημένης σελιδοποίησης όπου κάθε σελίδα αποτελείται μόνο από μία λέξη.

Ανακεφαλαίωση

Η μνήμη ενός υπολογιστή αποτελείται από τις μονάδες του που χρησιμεύουν στην αποθήκευση προγραμμάτων και δεδομένων. Η κύρια μνήμη (ή απλώς μνήμη) κρατά τα προγράμματα και τα δεδομένα για όσο χρόνο ο υπολογιστής λειτουργεί και αναφέρεται επίσης και ως RAM. Η δευτερεύουσα μνήμη (δίσκοι, CD-ROM κλπ.) τα κρατά και όταν ο υπολογιστής δε λειτουργεί. Το πρόβλημα οργάνωσης της μνήμης είναι πολύ σημαντικό κατά το σχεδιασμό ενός ΛΣ, ιδιαίτερα αν ο υπολογιστής εκτελεί πολλές διεργασίες και πρέπει να κάνει ανταλλαγή διεργασιών μεταξύ κύριας και δευτερεύουσας μνήμης, επειδή δε χωρούν όλα τα απαραίτητα προγράμματα και δεδομένα στην κύρια μνήμη. Οι εικονικές διευθύνσεις βοηθούν στην παράλληλη εκτέλεση πολλών διεργασιών χωρίς η μια να παρεμβαίνει στις θέσεις μνήμης που χρησιμοποιεί η άλλη. Για τη μετάφραση εικονικών διευθύνσεων σε φυσικές, πραγματικές διευθύνσεις μνήμης, υπάρχουν διάφορες τεχνικές που χρησιμοποιούν έναν πίνακα αντιστοίχισης. Η πιο δημοφιλής από αυτές είναι η σελιδοποίηση, που αντιστοιχίζει σελίδες μνήμης σταθερού μεγέθους.

Γλωσσάριο όρων

Ανταλλαγή	*Swapping*
Δευτερεύουσα / Περιφερειακή Μνήμη	*Secondary Memory*
Εικονική Διεύθυνση	*Virtual Address*
Εικονική Μνήμη	*Virtual Memory*
Ενότητα	*Block – Frame*
Εσωτερικός Κατακερματισμός	*Internal Fragmentation*
Δυναμική Κατανομή	*Dynamic Allocation*
Κατατμημένη Σελιδοποίηση	*Segmented Paging*
Κατάτμηση	*Segmentation*
Καταχωρητής Πίνακα Σελίδων	*Page Table Register*
Κύρια / Κεντρική Μνήμη	*Main Memory*
Λέξη	*Word*
Μνήμη ROM	*Read Only Memory*
Μνήμη Τυχαίας Προσπέλασης	*Random Access Memory - RAM*
Προσπέλαση Μνήμης	*Memory Access*
Σελίδα	*Page*
Σελιδοποίηση	*Paging*
Στατική Κατανομή	*Static Allocation*
Τμήμα	*Segment*
Χώρος Απολύτων Διευθύνσεων	*Absolute Address Space*
Χώρος Εικονικών Διευθύνσεων	*Virtual Address Space*
Χώρος Πραγματικών Διευθύνσεων	*Real Address Space*
Χώρος Φυσικών Διευθύνσεων	*Physical Address Space*

Ερωτήσεις

? Ποια είναι η βασική διαφορά της κύριας από τη δευτερεύουσα μνήμη; Είναι και οι δύο απολύτως απαραίτητες για τη λειτουργία του υπολογιστή;

? Ποια είναι τα μειονεκτήματα της ανταλλαγής;

? Η εικονική μνήμη έχει φυσική υπόσταση; Αν όχι, τι είναι ο εικονικός χώρος διευθύνσεων;

? Πώς μεταφράζεται μία εικονική διεύθυνση σε φυσική όταν το ΛΣ χρησιμοποιεί σελιδοποίηση για τη διαχείριση της εικονικής μνήμης;

Μάθημα 9.2 Τεχνικές Διαχείρισης Εικονικής Μνήμης

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να περιγράψει τις διάφορες τεχνικές που χρησιμοποιούν τα ΛΣ για να διαχειριστούν τη μνήμη.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να αναφέρεις τα προβλήματα που λύνει η διαχείριση της εικονικής μνήμης*
- *Να εξηγείς πότε γίνεται μεταφορά σελίδων προς την κύρια μνήμη*
- *Να εξηγείς ποια δεδομένα αντικαθίστανται κατά τη μεταφορά αυτή*
- *Να περιγράφεις πώς επιλέγονται οι διευθύνσεις της φυσικής μνήμης όπου τοποθετούνται νέα δεδομένα*

Όπως είδαμε στα προηγούμενα μαθήματα, ο χώρος εικονικών διευθύνσεων μνήμης που χρησιμοποιεί κάθε διεργασία, είναι αρκετά μεγαλύτερος από το χώρο των φυσικών διευθύνσεων. Έτσι, διάφορες εικονικές διευθύνσεις για μια διεργασία δεν αντιστοιχούν στην κύρια, αλλά στη δευτερεύουσα μνήμη. Για να χρησιμοποιηθούν όμως από τη διεργασία, πρέπει τα αντίστοιχα δεδομένα να μεταφερθούν στην κύρια μνήμη, αφού κάποια άλλα δεδομένα μεταφερθούν από την κύρια στη δευτερεύουσα μνήμη και να αλλάξουν οι αντιστοιχίσεις εικονικών και φυσικών διευθύνσεων ώστε να απεικονίζουν σωστά τη νέα κατάσταση.

Μια διεύθυνση δε μεταφέρεται μόνη της από τη δευτερεύουσα στην κύρια μνήμη, αλλά μαζί με τις γειτονικές της, γιατί αυτές έχουν πολύ αυξημένη πιθανότητα να χρησιμοποιηθούν στη συνέχεια από τη διεργασία, εξαιτίας της τοπικότητας αναφοράς των προγραμμάτων. Έτσι μεταφέρεται ολόκληρη η σελίδα μνήμης στην οποία περιέχεται η διεύθυνση αυτή. Η τοπικότητα της αναφοράς είναι μία ιδιότητα των προγραμμάτων που χρησιμοποιείται γενικά στο σύστημα της μνήμης ενός υπολογιστή, όπως είδαμε και στο μάθημα για τη λανθάνουσα μνήμη.

Η διαχείριση της εικονικής μνήμης από το ΛΣ πρέπει να δίνει απάντηση σε τρεις ουσιαστικές ερωτήσεις:

- ✔ Πότε πρέπει να γίνεται **μεταφορά** δεδομένων από τη δευτερεύουσα στην κύρια μνήμη;
- ✔ Ποια δεδομένα θα **αντικατασταθούν** και θα επιστρέψουν στη δευτερεύουσα μνήμη;
- ✔ Σε ποιες διευθύνσεις της φυσικής μνήμης θα **τοποθετηθούν** νέα δεδομένα;

Τα δυο πρώτα ερωτήματα αφορούν τα συστήματα σελιδοποίησης και τα συστήματα κατάτμησης, ενώ το τρίτο αφορά κυρίως τα συστήματα κατάτμησης όπου τα τμήματα των δεδομένων έχουν διαφορετικά μεγέθη και πρέπει να βρεθεί μια συνεχόμενη περιοχή φυσικής μνήμης όπου να χωρούν.

Μεταφορά σελίδων

Όπως γνωρίζουμε, το ΛΣ επιδιώκει να εξυπηρετεί πολλές διεργασίες ταυτόχρονα, αλλά έτσι δεν είναι δυνατόν να βρίσκονται συγχρόνως φορτωμένες στην κύρια μνήμη όλες οι σελίδες για κάθε διεργασία. Ανάλογα με τις ανάγκες της κάθε διεργασίας γίνεται μεταφορά σελίδων από και προς την κύρια μνήμη· η στρατηγική μεταφοράς που εφαρμόζει το ΛΣ θα έχει σαν σκοπό να αποφασίσει πότε θα γίνεται μεταφορά σελίδων ώστε να εξοικονομείται χρόνος. Δυο είναι οι κύριες στρατηγικές μεταφοράς:

1. Η πιο συνηθισμένη στρατηγική είναι η *σελιδοποίηση με αίτηση* (demand paging), σύμφωνα με την οποία μια σελίδα μεταφέρεται στην κύρια μνήμη μόνο όταν ζητηθεί κάποια διεύθυνση που περιέχεται σε αυτή και διαπιστωθεί ότι η σελίδα βρίσκεται στη δευτερεύουσα μνήμη. Η στρατηγική αυτή εξασφαλίζει ότι μια σελίδα θα μεταφερθεί από τη δευτερεύουσα στην κύρια μνήμη μόνο όταν αυτό είναι απαραίτητο. Από την άλλη όμως, μια διεργασία πρέπει να περιμένει κάθε φορά που θα ζητήσει την προσπέλαση μιας εικονικής διεύθυνσης που αντιστοιχεί στη δευτερεύουσα μνήμη. Μάλιστα, όσο περισσότερες διεργασίες εκτελούνται ταυτόχρονα από το ΛΣ, τόσο πιο συχνά πρέπει να περιμένουν για τη μεταφορά των σελίδων τους στην κύρια μνήμη.
2. Αν μπορούν να προβλεφθούν σε ικανοποιητικό βαθμό οι σελίδες που θα προσπελάσει μια διεργασία πριν να τις προσπελάσει, τότε εφαρμόζεται η στρατηγική της *σελιδοποίησης με πρόβλεψη* (anticipatory paging). Η στρατηγική αυτή προβλέπει ποιες σελίδες θα προσπελάσει βραχυπρόθεσμα μια διεργασία και τις μεταφέρει εκ των προτέρων στην κύρια μνήμη· έτσι η διεργασία δε χρειάζεται να περιμένει για να προσπελάσει οποιαδήποτε διεύθυνση. Είναι όμως πιθανόν ότι θα μεταφερθούν στην κύρια μνήμη σελίδες που δε θα χρειαστούν και θα καταλαμβάνουν χώρο χωρίς λόγο. Επίσης, ο υπολογισμός των σελίδων που θα μεταφερθούν στην κύρια μνήμη απαιτεί κάποιο χρόνο και επιβαρύνει το ΛΣ.

Από τις δυο στρατηγικές συχνότερα χρησιμοποιείται η σελιδοποίηση με αίτηση, γιατί η υλοποίησή της είναι απλούστερη και επιπλέον διότι η πρόβλεψη των σελίδων που θα χρειαστεί στο άμεσο μέλλον μια διεργασία είναι στην πλειοψηφία των περιπτώσεων δύσκολη.

Αντικατάσταση σελίδων

Όταν το ΛΣ αποφασίσει ότι θα μεταφέρει μια σελίδα από τη δευτερεύουσα στην κύρια μνήμη, πρέπει να επιλέξει μια σελίδα στην κύρια μνήμη που θα αντικατασταθεί και θα μεταφερθεί στη δευτερεύουσα μνήμη. Στόχος του ΛΣ είναι να επιλέξει για μεταφορά στη δευτερεύουσα μνήμη μια σελίδα που έχει μικρή πιθανότητα να χρησιμοποιηθεί σύντομα.

> Σε έναν υπολογιστή με κύρια μνήμη μεγέθους 4 ενοτήτων εκτελείται η διεργασία δ_α, η οποία απαιτεί 7 σελίδες μνήμης (α_0, α_1, α_2, α_3, α_4, α_5, και α_6). Όταν ξεκινά η διεργασία, στην κύρια μνήμη δεν είναι αποθηκευμένη καμιά σελίδα της. Στη συνέχεια η ζήτηση σελίδων από τη διεργασία είναι: α_0, α_2, α_6, α_5, α_2, α_1, α_0, α_3, α_4, α_0, α_6. Υποθέτουμε ότι οι σελίδες ζητούνται σε διαδοχικές χρονικές στιγμές, δηλαδή τη χρονική στιγμή 0 ζητείται η σελίδα α_0, τη χρονική στιγμή 1 ζητείται η σελίδα α_2 κ.ο.κ.

Στη μία στρατηγική, αυτή της *αντικατάστασης με βάση το χρόνο παραμονής* (First In First Out, FIFO), επιλέγεται η σελίδα που βρίσκεται για το μεγαλύτερο χρονικό διάστημα στην κύρια μνήμη και αντικαθίσταται με τη νέα. Για να εντοπίζεται η παλαιότερη σελίδα, κάθε φορά που μια σελίδα μεταφέρεται στην κύρια μνήμη σημειώνεται με τον τρέχοντα χρόνο (από το ρολόι του υπολογιστή). **Η σελίδα που είναι σημειωμένη με το μικρότερο χρόνο είναι η παλαιότερη.**

Τις τέσσερις πρώτες χρονικές στιγμές (0, 1, 2, 3), που ζητούνται οι σελίδες α_0, α_2, α_6 και α_5 αντίστοιχα, οι τέσσερις σελίδες της μνήμη είναι κενές, οπότε οι σελίδες τοποθετούνται απευθείας. Στη συνέχεια, οι αντικαταστάσεις των σελίδων φαίνονται στο ακόλουθο σχήμα που παριστάνει τον πίνακα σελίδων. Στη στήλη 📖 φαίνονται οι σελίδες της εικονικής μνήμης με την αντίστοιχη ενότητα κύριας μνήμης (ή «-» αν η σελίδα δε βρίσκεται στην κύρια μνήμη), και στη στήλη 🕑 φαίνεται ο χρόνος με τον οποίο έχουν σημειωθεί όσες σελίδες βρίσκονται στην κύρια μνήμη. Για κάθε χρονική στιγμή, δίνεται η σελίδα που ζητά η διεργασία και εξηγείται η αντικατάσταση που θα γίνει, αν είναι απαραίτητη. Η σελίδα που θα αντικατασταθεί κάθε φορά είναι σκιασμένη.

Τη χρονική στιγμή 4 προσπελάζεται η σελίδα α_2, η οποία είναι ήδη φορτωμένη στη μνήμη, οπότε δε γίνεται καμία αντικατάσταση.

	📖	🕑
α_0	0	0
α_1	-	-
α_2	1	1
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_2** ⇓⇓

Τη χρονική στιγμή 5 η διεργασία ζητά τη σελίδα α_1. Η παλαιότερη σελίδα είναι η α_0, η οποία θα αντικατασταθεί από την α_1 και θα σημειωθεί με το χρόνο 5.

	📖	🕑
α_0	0	0
α_1	-	-
α_2	1	1
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_1** ⇓⇓

Τη χρονική στιγμή 6 προσπελάζεται πάλι η α_0, που δε βρίσκεται πλέον στη μνήμη. Μεταφέρεται έτσι η α_2, που ήταν η παλαιότερη, στη δευτερεύουσα μνήμη, και η α_0 παίρνει τη θέση της.

	📖	🕑
α_0	-	-
α_1	0	5
α_2	1	1
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_0** ⇓⇓

Τη χρονική στιγμή 7 ζητείται η σελίδα α_3, η οποία θα αντικατα-στήσει τη α_6 που ήταν η παλαιότερη. Η α_3 σημειώνεται με το χρόνο 7.

	📖	🕑
α_0	1	6
α_1	0	5
α_2	-	-
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

Συνολικά για τις 11 προσπελάσεις σελίδων έγιναν 5 αντικαταστάσεις.

	📖	🕑
α_0	1	6
α_1	-	-
α_2	-	-
α_3	2	7
α_4	3	8
α_5	-	-
α_6	0	10

⇑⇑ **προσπέλαση της α_6** ⇑⇑

Τη χρονική στιγμή 10 ζητείται η σελίδα α_6 · η παλαιότερη σελίδα είναι η α_1 με χρόνο 5.

	📖	🕑
α_0	1	6
α_1	0	5
α_2	-	-
α_3	2	7
α_4	3	8
α_5	-	-
α_6	-	-

⇑⇑ **προσπέλαση της α_0** ⇑⇑

Τη χρονική στιγμή 9, προσπελάζεται πάλι η σελίδα α_0 που ήδη βρίσκεται στην κύρια μνήμη, και έτσι δε γίνεται καμία αντικατάσταση.

	📖	🕑
α_0	1	6
α_1	0	5
α_2	-	-
α_3	2	7
α_4	3	8
α_5	-	-
α_6	-	-

⇑⇑ **προσπέλαση της α_4** ⇑⇑

Τη χρονική στιγμή 8, η διεργασία ζητά τη σελίδα α_4. Αυτή θα τοποθετηθεί αντί για τη α_5 που είχε το μικρότερο χρόνο εισόδου στην κύρια μνήμη.

	📖	🕑
α_0	1	6
α_1	0	5
α_2	-	-
α_3	2	7
α_4	-	-
α_5	3	3
α_6	-	-

⇓⇒⇒⇒ **προσπέλαση της α_3** ⇒⇒⇒⇑

Η δεύτερη στρατηγική είναι αυτή της *αντικατάστασης με βάση το χρόνο της τελευταίας προσπέλασης* (Least Recently Used, LRU), οπότε επιλέγεται για αντικατάσταση εκείνη η σελίδα που έχει τον περισσότερο χρόνο να προσπελαστεί. Για να είναι αυτό δυνατόν, πρέπει **κάθε φορά** που μια σελίδα προσπελάζεται, είτε μεταφέρεται στην κύρια μνήμη είτε ήδη βρίσκεται εκεί, να σημειώνεται με τον τρέχοντα χρόνο του υπολογιστή*. Όταν πρέπει να αντικατασταθεί μια σελίδα, το ΛΣ διαλέγει εκείνη που είναι σημειωμένη με το μικρότερο (δηλαδή τον παλαιότερο) χρόνο.

Και με την τεχνική της αντικατάστασης με βάση το χρόνο προσπέλασης, οι τέσσερις πρώτες προσπελάσεις σελίδων, οπότε ζητούνται οι σελίδες α_0, α_2, α_6 και α_5, έχουν σαν αποτέλεσμα την απευθείας τοποθέτηση των σελίδων στην κύρια μνήμη. Στη συνέχεια, οι αντικαταστάσεις των σελίδων γίνονται ως εξής:

Τη χρονική στιγμή 4 προσπελάζεται η σελίδα α_2, η οποία είναι ήδη φορτωμένη στη μνήμη, οπότε δε γίνεται καμία αντικατάσταση, αλλά μόνο σημείωσή της με τον τρέχοντα χρόνο.

α_0	0	0
α_1	-	-
α_2	1	1
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_2** ⇓⇓

Τη χρονική στιγμή 5 η διεργασία α ζητά τη σελίδα α_1. Η παλαιότερα χρησιμοποιημένη σελίδα είναι η α_0, η οποία θα αντικατασταθεί από την α_1 και θα σημειωθεί με το χρόνο 5.

α_0	0	0
α_1	-	-
α_2	1	4
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_1** ⇓⇓

Τη χρονική στιγμή 6 προσπελάζεται πάλι η α_0, που δε βρίσκεται πλέον στη μνήμη. Μεταφέρεται έτσι η α_6, που είχε το μικρότερο χρόνο προσπέλασης, στη δευτερεύουσα μνήμη.

α_0	-	-
α_1	0	5
α_2	1	4
α_3	-	-
α_4	-	-
α_5	3	3
α_6	2	2

⇓⇓ **προσπέλαση της α_0** ⇓⇓

Τη χρονική στιγμή 7 ζητείται η σελίδα α_3, η οποία θα αντικατα-στήσει τη α_5 που έχει τον παλαιότερο χρόνο προσπέλασης. Η α_3 σημειώνεται με το χρόνο 7.

α_0	2	6
α_1	0	5
α_2	1	4
α_3	-	-
α_4	-	-
α_5	3	3
α_6	-	-

⇓⇒⇒⇒ **προσπέλαση της α_3** ⇒⇒⇒⇑

Συνολικά για τις 11 προσπελάσεις σελίδων έγιναν και με την τεχνική αυτή 5 αντικαταστάσεις, αλλά διαφορετικές από ό,τι με την τεχνική του χρόνου παραμονής.

α_0	2	9
α_1	-	-
α_2	-	-
α_3	3	7
α_4	1	8
α_5	-	-
α_6	0	10

⇑⇑ **προσπέλαση της α_6** ⇑⇑

Τη χρονική στιγμή 10 ζητείται η σελίδα α_6 · η λιγότερο πρόσφατα χρησιμοποιημένη σελίδα είναι η α_1 με χρόνο προσπέλασης 5.

α_0	2	9
α_1	0	5
α_2	-	-
α_3	3	7
α_4	1	8
α_5	-	-
α_6	-	-

⇑⇑ **προσπέλαση της α_0** ⇑⇑

Τη χρονική στιγμή 9, προσπελάζεται πάλι η σελίδα α_0 που ήδη βρίσκεται στην κύρια μνήμη, και έτσι απλά η σελίδα σημειώνεται με το νέο χρόνο του υπολογιστή.

α_0	2	6
α_1	0	5
α_2	-	-
α_3	3	7
α_4	1	8
α_5	-	-
α_6	-	-

⇑⇑ **προσπέλαση της α_4** ⇑⇑

Τη χρονική στιγμή 8, η διεργασία ζητά τη σελίδα α_4. Αυτή θα τοποθετηθεί αντί για την α_2 που είχε το μικρότερο χρόνο προσπέλασης.

α_0	2	6
α_1	0	5
α_2	1	4
α_3	3	7
α_4	-	-
α_5	-	-
α_6	-	-

---

* Ενώ στην τεχνική FIFO μια σελίδα σημειώνεται *μόνο* όταν μεταφέρεται στην κύρια μνήμη.

Τοποθέτηση σελίδων

Σε ένα σύστημα σελιδοποίησης είναι πολύ εύκολο να επιλεγεί η ενότητα της κύριας μνήμης όπου θα τοποθετηθεί μια σελίδα εικονικών διευθύνσεων, γιατί όλες έχουν το ίδιο μέγεθος. Στα συστήματα κατάτμησης όμως, όπου τα τμήματα έχουν διαφορετικά μεγέθη, κάθε φορά που μεταφέρεται ένα νέο τμήμα στην κύρια μνήμη, πρέπει να βρεθεί μια κατάλληλη συνεχόμενη περιοχή της κύριας μνήμης η οποία να το χωρά. Η επιλογή της περιοχής αυτής γίνεται με βάση δυο κριτήρια:

- Η αναζήτηση της κατάλληλης ελεύθερης περιοχής πρέπει να γίνεται γρήγορα, για να μην προκαλούνται μεγάλες χρονικές επιβαρύνσεις.
- Η τοποθέτηση του τμήματος στην κύρια μνήμη πρέπει να παρεμποδίζει όσο το δυνατόν λιγότερο την τοποθέτηση επομένων τμημάτων.

Οι στρατηγικές τοποθέτησης σελίδων που χρησιμοποιούνται περισσότερο είναι η στρατηγική του *πρώτου ταιριάσματος* (first fit), η στρατηγική του *καλύτερου ταιριάσματος* (best fit) και η στρατηγική του *χειρότερου ταιριάσματος* (worst fit).

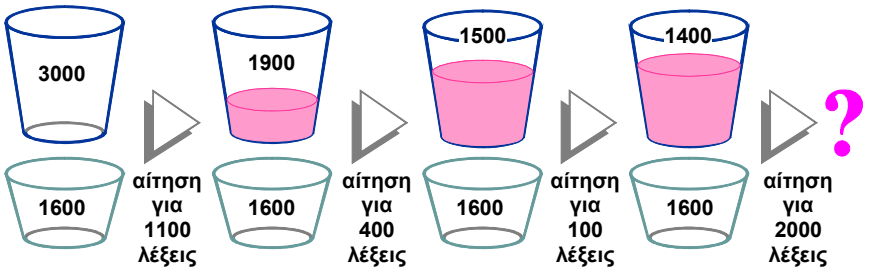
Όλες αυτές οι στρατηγικές προϋποθέτουν ότι το ΛΣ κρατά ένα κατάλογο με τα ελεύθερα τμήματα φυσικής μνήμης και το μέγεθος του καθενός από αυτά. Κάθε φορά που επιλέγεται ένα από τα ελεύθερα τμήματα για να τοποθετηθεί σε αυτό ένα τμήμα εικονικής μνήμης, αφαιρείται από τον κατάλογο. Αν τυχόν το τμήμα της εικονικής μνήμης είναι μικρότερο από αυτό της κύριας μνήμης, η μνήμη που περισσεύει επιστρέφεται στον κατάλογο των ελευθέρων τμημάτων.

Στρατηγική του πρώτου ταιριάσματος: Σαρώνεται ο κατάλογος ελεύθερων τμημάτων μέχρι να βρεθεί το πρώτο τμήμα που είναι αρκετά μεγάλο για να χωρά το νέο τμήμα εικονικής μνήμης. Ό,τι περισσεύει επιστρέφεται στον κατάλογο σαν ένα μικρότερο ελεύθερο τμήμα. Η στρατηγική αυτή είναι η πιο γρήγορη, γιατί δε σαρώνει ολόκληρο τον κατάλογο ελευθέρων τμημάτων, αλλά μπορεί να δημιουργήσει πολλά μικρά τμήματα ελεύθερης μνήμης που δεν μπορούν να χρησιμοποιηθούν.

Στρατηγική του καλύτερου ταιριάσματος: Η στρατηγική αυτή εξετάζει όλα τα ελεύθερα τμήματα, και επιλέγει το μικρότερο ελεύθερο κομμάτι φυσικής μνήμης που χωρά το τμήμα εικονικής μνήμης. Επειδή πρέπει να εξετάσει όλα τα ελεύθερα τμήματα είναι μάλλον αργή, και επιπλέον αφήνει πολύ μικρά κομμάτια ελεύθερης μνήμης τα οποία δεν μπορούν να χρησιμοποιηθούν λόγω του μικρού τους μεγέθους.

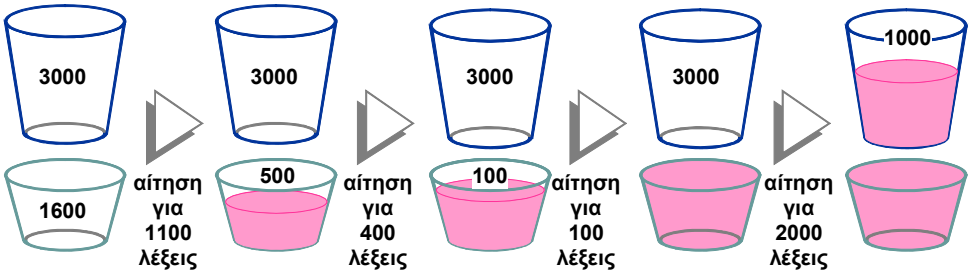
Στρατηγική του χειρότερου ταιριάσματος: Επιλέγει κάθε φορά το μεγαλύτερο κομμάτι μνήμης (εφόσον υπάρχει κάποιο αρκετά μεγάλο ώστε να χωρά το τμήμα που θέλουμε να τοποθετήσουμε). Και αυτή η μέθοδος πρέπει να εξετάσει όλόκληρο τον κατάλογο, εκτός και αν αυτός είναι ταξινομημένος κατά μέγεθος σε φθίνουσα σειρά.

Αρχικά ο κατάλογος ελεύθερων τμημάτων περιέχει δυο τμήματα φυσικής μνήμης, ένα με μέγεθος 3000 και ένα με μέγεθος 1600 λέξεις. Στη συνέχεια ζητούνται διαδοχικά τέσσερα τμήματα μνήμης, με μεγέθη 1100, 400, 100 και 2000 λέξεις, αντίστοιχα.

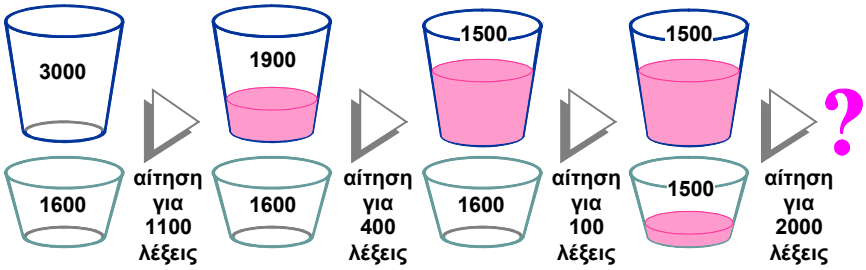


Στρατηγική του πρώτου ταιριάσματος

Στρατηγική του καλύτερου ταιριάσματος



Στρατηγική του χειρότερου ταιριάσματος



Στο παράδειγμα αυτό η στρατηγική του καλύτερου ταιριάσματος είναι καλύτερη από τις άλλες δύο, οι οποίες δεν μπόρεσαν να ικανοποιήσουν την τελευταία αίτηση μνήμης. Σε άλλα παραδείγματα όμως μπορεί αυτή να μην είναι τόσο καλή, ενώ κάποια άλλη (π.χ. η η στρατηγική του πρώτου ταιριάσματος) μπορεί να αποδειχθεί αποδοτικότερη.

Ανακεφαλαίωση

Στην αντιστοίχιση εικονικής με φυσική μνήμη, τρία είναι τα ζητήματα που πρέπει να αντιμετωπιστούν: αν η μεταφορά της μνήμης θα γίνεται με αίτηση ή με πρόβλεψη, πώς θα επιλέγεται η σελίδα που θα αντικατασταθεί και πού θα τοποθετηθεί ένα νέο τμήμα μνήμης. Για την αντικατάσταση υπάρχουν οι τεχνικές αντικατάστασης με βάση το χρόνο προσπέλασης ή το χρόνο πρόσβασης, και για την τοποθέτηση χρησιμοποιούνται οι στρατηγικές του πρώτου, του καλύτερου ή του χειρότερου ταιριάσματος.

Γλωσσάριο όρων

Αντικατάσταση με Βάση το Χρόνο Παραμονής	*First In First Out - FIFO*
Αντικατάσταση με Βάση το Χρόνο της Τελευταίας Προσπέλασης	*Least Recently Used - LRU*
Σελιδοποίηση με Αίτηση	*Demand Paging*
Σελιδοποίηση με Πρόβλεψη	*Anticipatory Paging*
Στρατηγική του Καλύτερου Ταιριάσματος	*Best Fit*
Στρατηγική του Πρώτου Ταιριάσματος	*First Fit*
Στρατηγική του Χειρότερου Ταιριάσματος	*Worst Fit*

Ερωτήσεις

? Ποια είναι τα πλεονεκτήματα και τα μειονεκτήματα της μεταφοράς σελίδων με πρόβλεψη;

? Περίγραψε τη μέθοδο αντικατάστασης σελίδων με βάση το χρόνο τελευταίας πρόσβασης.

? Ποια νομίζεις ότι είναι η βασική ιδέα της στρατηγικής του χειρότερου ταιριάσματος;

Τι μάθαμε σε αυτό το κεφάλαιο

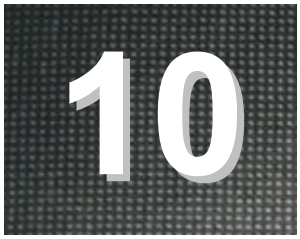
- Οι εικονικές διευθύνσεις βοηθούν στην παράλληλη εκτέλεση πολλών διεργασιών χωρίς η μια να παρεμβαίνει στις θέσεις μνήμης που χρησιμοποιεί η άλλη.
- Η πιο δημοφιλής τεχνική αντιστοίχισης εικονικών διευθύνσεων σε φυσικές είναι η σελιδοποίηση.
- Στην αντιστοίχιση εικονικής με φυσική μνήμη, τρία είναι τα ζητήματα που πρέπει να αντιμετωπιστούν: αν η μεταφορά της μνήμης θα γίνεται με αίτηση ή με πρόβλεψη, πώς θα επιλέγεται η σελίδα που θα αντικατασταθεί και πού θα τοποθετηθεί ένα νέο τμήμα μνήμης.

Βιβλιογραφία – Πηγές

📖 Παπακωσταντίνου Γ., Ν. Μπιλάλη, Π. Τσανάκα, Λειτουργικά Συστήματα: Αρχές Λειτουργίας, Συμμετρία, 1989.

📖 Silberschatz A., Peterson J., Galvin P., Operating System Concepts, Addison - Wesley, 1991.

Κεφάλαιο



Διαχείριση Αρχείων στα Λειτουργικά Συστήματα

Σκοπός του κεφαλαίου αυτού είναι να περιγράψει τη διαχείριση των αρχείων από ένα Λειτουργικό Σύστημα, δηλαδή των δεδομένων που κρατούνται σε κάποια μονάδα δευτερεύουσας μνήμης.

Όταν ολοκληρώσεις το κεφάλαιο αυτό, θα μπορείς:

- Να ορίζεις το σύστημα αρχείων.
- Να αναφέρεις τις ιδιότητες των αρχείων και τις λειτουργίες που τα επηρεάζουν.
- Να εξηγείς τι είναι κατάλογος και διαδρομή και πώς τα χρησιμοποιούμε.

Μαθήματα

10.1 Σύστημα Αρχείων

Μάθημα 10.1 Σύστημα Αρχείων

Σκοπός του μαθήματος

Σκοπός του μαθήματος αυτού είναι να εξηγήσει τη χρησιμότητα και τη λειτουργία του συστήματος αρχείων.

Τι θα μάθεις;

Όταν ολοκληρώσεις το μάθημα αυτό, θα μπορείς:

- *Να ορίζεις το σύστημα αρχείων*
- *Να αναφέρεις τις ιδιότητες των αρχείων και τις λειτουργίες που τα επηρεάζουν*
- *Να εξηγείς τι είναι κατάλογος και διαδρομή και πώς τα χρησιμοποιούμε*

Ένα *αρχείο* (file) αποτελείται από ένα σύνολο δεδομένων αποθηκευμένων στη δευτερεύουσα μνήμη. Τα δεδομένα αυτά μπορούν να χρησιμοποιούνται από προγράμματα ή να περιέχουν πληροφορίες που έχουν καταγραφεί από κάποιο χρήστη και τον ενδιαφέρουν. Η βασική ιδιότητα των αρχείων, η οποία τα διαφοροποιεί από τις διάφορες πληροφορίες που είναι αποθηκευμένες στην κύρια μνήμη του υπολογιστή, είναι ότι είναι *μόνιμα*, δηλαδή εξακολουθούν να υπάρχουν και όταν η διεργασία που τα δημιούργησε έχει τερματιστεί ή ακόμα και όταν ο υπολογιστής είναι εκτός λειτουργίας.

Το *Σύστημα Αρχείων* (file system) είναι το τμήμα εκείνο του ΛΣ που αναλαμβάνει να οργανώσει και να διαχειριστεί τα δεδομένα των αρχείων, δίνοντας τη δυνατότητα στους χρήστες:

- να αναφέρονται στα αρχεία με κάποιο συμβολικό όνομα
- να τα ομαδοποιούν σε μια λογική δομή ανάλογα με τη φύση των δεδομένων που περιέχουν και τη χρησιμότητά τους
- να εκτελούν διάφορες σύνθετες λειτουργίες σε ένα ολόκληρο αρχείο
- να χρησιμοποιούν τα δεδομένα των αρχείων χωρίς να ενδιαφέρονται πού ακριβώς βρίσκονται αυτά στη δευτερεύουσα μνήμη

Για να αναφερθεί ο χρήστης ή μια διεργασία σε ένα αρχείο και να μπορεί να χρησιμοποιήσει τα περιεχόμενά του, δίνεται σε αυτό ένα *όνομα αρχείου* (file name). Με τη βοήθεια του ονόματος αυτού μπορεί ο χρήστης να δώσει εντολές προς το ΛΣ ζητώντας π.χ. την αποστολή των δεδομένων του αρχείου στον εκτυπωτή.

Το σύστημα αρχείων ενός υπολογιστή είναι αντίστοιχο με τη μέθοδο που χρησιμοποιεί ο ζαχαροπλάστης για να οργανώνει τις συνταγές του. Κάθε συνταγή είναι ένα αρχείο, με όνομα τον τίτλο της συνταγής.

Το όνομα αρχείου αποτελείται από μια ακολουθία χαρακτήρων, η οποία ανάλογα με το συγκεκριμένο ΛΣ μπορεί να περιέχει γράμματα, αριθμητικά ψηφία και ειδικά σύμβολα όπως «_», «.», «!». Κάθε ΛΣ έχει το δικό του σύνολο κανόνων για το μέγιστο μήκος του ονόματος ενός αρχείου, τα ειδικά σύμβολα που επιτρέπεται ή όχι να περιέχει, το αν τα κεφαλαία και τα πεζά γράμματα είναι ισοδύναμα κλπ.

Συνήθως το όνομα του αρχείου είναι μια μικρή «περιγραφή» των δεδομένων τα οποία περιέχει. Π.χ. το αρχείο με όνομα «letter» κατά πάσα πιθανότητα περιέχει το κείμενο κάποιας επιστολής. Μια σύμβαση, που χρησιμοποιείται από αρκετά προγράμματα και ΛΣ έτσι ώστε να δίνει το όνομα πιο πολλές πληροφορίες για τα περιεχόμενα του αρχείου είναι αυτή της *προέκτασης ονόματος* (file name extension). Αυτή είναι το τελευταίο τμήμα του ονόματος του αρχείου, που χωρίζεται από το υπόλοιπο όνομα με μια τελεία. Η προέκταση μπορεί να έχει περιορισμένο μήκος ή όχι, και περιέχει ένα συνδυασμό γραμμάτων ή μια λέξη που δηλώνει τον τύπο δεδομένων του αρχείου.

> Ένα αρχείο που έχει την προέκταση «txt», π.χ. το letter.txt κατά πάσα πιθανότητα περιέχει το κείμενο (text) μιας επιστολής. Η προέκταση αυτή ερμηνεύεται από το χρήστη, όμως υπάρχουν άλλες που ερμηνεύονται από προγράμματα ή το ίδιο το ΛΣ. Ένα αρχείο με προέκταση «lib» μπορεί να περιέχει κάποια βιβλιοθήκη προγραμμάτων που καλεί το ΛΣ, και ένα άλλο με προέκταση «c» μπορεί να αναγνωρίζεται από ένα μεταφραστή της γλώσσας προγραμματισμού C ως ένα πρόγραμμα γραμμένο σε C.

Συχνά τα ΛΣ προκαθορίζουν ορισμένες προεκτάσεις αρχείων οι οποίες έχουν ειδική σημασία, όπως προεκτάσεις για τα προγράμματα που μπορούν να εκτελεστούν, δηλαδή τα *εκτελέσιμα* (executable) αρχεία. Επίσης καθορίζουν και διάφορους άλλους τύπους αρχείων, οι οποίοι όμως δε δηλώνονται από το όνομα του αρχείου. Ένας τέτοιος τύπος αρχείου είναι ο *κατάλογος* ή *ευρετήριο* (directory). Αυτός περιέχει μια καταγραφή των αρχείων που βρίσκονται στη συγκεκριμένη μονάδα δευτερεύουσας μνήμης, μαζί με διάφορες πληροφορίες για το πού μπορεί το ΛΣ να βρει τα περιεχόμενα του καθενός αρχείου.

Χαρακτηριστικά των αρχείων

Το σύστημα αρχείων του ΛΣ χρειάζεται διάφορες πληροφορίες για κάθε αρχείο, οι οποίες ονομάζονται *χαρακτηριστικά* (attributes - properties). Το βασικό χαρακτηριστικό του αρχείου είναι βέβαια το όνομά του. Υπάρχουν όμως και άλλες απαραίτητες πληροφορίες, μερικές από τις οποίες μπορεί να είναι:

- Το μέγεθος του αρχείου
- Ο χρήστης που δημιούργησε το αρχείο
- Ο τύπος του αρχείου
- Ο τρέχων χρήστης-ιδιοκτήτης του
- Η ημερομηνία και ώρα δημιουργίας του αρχείου
- Η ημερομηνία και ώρα της τελευταίας πρόσβασης στο αρχείο
- Η ημερομηνία και ώρα της τελευταίας τροποποίησης των περιεχομένων του
- Μια ένδειξη αν τα δεδομένα του αρχείου μπορούν να αλλοιωθούν ή αυτό προορίζεται μόνο για ανάγνωση δεδομένων (read only)
- Μια ένδειξη αν πρόκειται για αρχείο του ΛΣ

- Οι χρήστες που δικαιούνται να το χρησιμοποιούν και με ποιο τρόπο: να διαβάζουν τα δεδομένα του, να τα αλλάζουν ή να τροποποιούν κάποια χαρακτηριστικά του
- Αν το αρχείο το χρησιμοποιούν από κοινού πολλές διεργασίες και ποιες.

Κάποιες από αυτές και άλλες πληροφορίες διατηρούνται από το ΛΣ στο *Τμήμα Ελέγχου Αρχείου, ΤΕΑ* (File Control Block, FCB) το οποίο είναι μέρος του συστήματος αρχείων.

Ιδιαίτερα σημαντικές πληροφορίες είναι εκείνες που αφορούν την πρόσβαση των χρηστών στα αρχεία, γιατί επηρεάζουν την ασφάλεια των δεδομένων του ΛΣ. Οι πληροφορίες αυτές καθορίζουν ποιες λειτουργίες μπορούν να εκτελεστούν και από ποιους χρήστες σε ένα αρχείο, και ονομάζονται *δικαιώματα προσπέλασης* (access rights).

> Για κάθε συνταγή του ο ζαχαροπλάστης κρατά διάφορες πληροφορίες: την πηγή της συνταγής, πότε την κατέγραψε, το χρόνο που απατείται για την εκτέλεσή της, το κόστος των υλικών, την ευκολία της κλπ.

Λειτουργίες που εκτελούνται στα αρχεία

Το τμήμα του ΛΣ που ασχολείται με το σύστημα αρχείων δίνει τη δυνατότητα στους χρήστες να εκτελέσουν διάφορες λειτουργίες στα αρχεία με δυο τρόπους:

1. Με κλήσεις ρουτινών συστήματος, οι οποίες γίνονται μέσα από τα προγράμματα του χρήστη
2. Με εντολές του ΛΣ, οι οποίες εισάγονται απευθείας από το χρήστη προς το ΛΣ. Οι εντολές αυτές συνήθως εκτελούν προγράμματα που χρησιμοποιούν κλήσεις συστήματος.

Οι κυριότερες λειτουργίες που παρέχονται από όλα τα ΛΣ είναι:

	Δημιουργία ενός νέου αρχείου. Αυτό αρχικά δεν περιέχει δεδομένα, δηλαδή το μέγεθός του είναι 0. Το νέο αρχείο, μετά τη δημιουργία του είναι «ανοικτό» και διαθέσιμο στη διεργασία που το δημιούργησε για να προσθέσει δεδομένα ξεκινώντας από την αρχή του.
	Διαγραφή ενός αρχείου από τη δευτερεύουσα μνήμη. Ανάλογα με το ΛΣ, τα περιεχόμενα του αρχείου μπορεί να χάνονται αυτομάτως ή να υπάρχει μια πιθανότητα να ανακτηθούν με ειδικά προγράμματα.
	Άνοιγμα ενός αρχείου από μια διεργασία, ώστε αυτή να έχει πρόσβαση στα περιεχόμενά του. Υπάρχουν διάφοροι τρόποι ανοίγματος ενός αρχείου: για ανάγνωση μόνο, για ανάγνωση και εγγραφή, για σειριακή ανάγνωση (δηλαδή μόνο από την αρχή προς το τέλος με τη σειρά). Ένα αρχείο μπορεί να ανοιχθεί ως μοιραζόμενο· στην περίπτωση αυτή πολλές διεργασίες μπορούν να έχουν συγχρόνως το ίδιο αρχείο ανοικτό, να διαβάζουν ή να γράφουν σε αυτό. Η προσπέλαση των δεδομένων του αρχείου από μια διεργασία γίνεται πάντα στο *τρέχον σημείο* του αρχείου. Όταν το αρχείο δημιουργείται ή ανοίγει, το τρέχον σημείο είναι η αρχή του.
	Κλείσιμο ενός αρχείου που είχε ανοιχθεί νωρίτερα από μια διεργασία. Η διεργασία δεν έχει πλέον πρόσβαση στα περιεχόμενα του αρχείου, και όποιες αλλαγές είχαν γίνει σ' αυτό και δεν είχαν οριστικοποιηθεί τώρα μπορούν να αποθηκευθούν μόνιμα.

	Ανάγνωση δεδομένων από ένα ανοικτό αρχείο. Η ανάγνωση γίνεται στο *τρέχον σημείο* του αρχείου. Κατόπιν το τρέχον σημείο μετακινείται αυτόματα στα αμέσως επόμενα δεδομένα από αυτά που διαβάστηκαν.
	Εγγραφή δεδομένων σε ένα ανοικτό αρχείο, στο τρέχον σημείο του. Μετά από την εγγραφή το τρέχον σημείο προωθείται αμέσως μετά από τα δεδομένα που γράφτηκαν· εκεί θα γίνει η επόμενη εγγραφή. Αν στο σημείο όπου έγινε η εγγραφή υπήρχαν δεδομένα, τότε αυτά χάνονται. Αν δεν υπήρχαν δεδομένα (π.χ. στο τέλος του αρχείου), τότε χρησιμοποιείται νέος χώρος στη δευτερεύουσα μνήμη για να καταγραφούν οι νέες πληροφορίες.
	Ανάγνωση ή αλλαγή των **χαρακτηριστικών** ενός αρχείου. Μια διεργασία μπορεί π.χ. να μάθει τον ιδιοκτήτη του αρχείου, την ημερομηνία δημιουργίας του κλπ. Δεν είναι δυνατόν όμως να αλλάξουν όλα τα χαρακτηριστικά του αρχείου, όπως το μέγεθος του αρχείου, το οποίο καθορίζεται έμμεσα από τα δεδομένα που περιέχει. Το κυριότερο χαρακτηριστικό ενός αρχείου που μπορεί να αλλάξει είναι το **όνομά** του.

Για να εκτελέσει μια διεργασία κάποια λειτουργία σε ένα αρχείο, πρέπει να διαθέτει τα ανάλογα δικαιώματα προσπέλασης σ' αυτό. Για να ανοίξει π.χ.ένα αρχείο για εγγραφή δεδομένων, η διεργασία (δηλαδή ο χρήστης στον οποίο ανήκει η διεργασία) πρέπει να έχει δικαίωμα εγγραφής στο αρχείο αυτό.

Για να εκτελέσει το ΛΣ όλες αυτές τις λειτουργίες χρησιμοποιεί δικά του αρχεία, τα οποία φυσικά δεν είναι προσπελάσιμα από τους χρήστες.

Κατάλογοι συστήματος αρχείων

Για να εντοπίζει εύκολα τις συνταγές του, ο ζαχαροπλάστης τις οργανώνει στα ράφια μιας βιβλιοθήκης και σε διαφορετικούς φακέλους. Στο πρώτο ράφι π.χ. έχει όλες τις συνταγές για κέικ, στο δεύτερο ράφι όλες τις συνταγές για γλυκά του κουταλιού κλπ. Στο ράφι με τα κέικ έχει ένα φάκελο για τα κέικ με αλεύρι, έναν για τα κέικ με σιμιγδάλι και έναν για τα υπόλοιπα. Τα κέικ με αλεύρι τα έχει χωρίσει στα απλά, αυτά με σοκολάτα, με ξηρούς καρπούς κλπ. Έτσι του είναι πολύ εύκολο να βρει τη συνταγή που θέλει, αλλά και να ταξινομήσει μια νέα συνταγή ανάλογα με το περιεχόμενό της.

Στο σύστημα αρχείων ενός υπολογιστή, όπου τα αρχεία συνήθως είναι πάρα πολλά, υπάρχει η ανάγκη για λογική οργάνωσή τους ανάλογα με τη λειτουργικότητα και το «θέμα» τους. Έτσι γίνεται κατανομή των αρχείων σε διαφορετικούς *καταλόγους* ή *ευρετήρια* (directories)· για να αναφερθεί ο χρήστης σε ένα αρχείο χρειάζεται το όνομα του αρχείου και τον κατάλογο όπου αυτό βρίσκεται. Οι κατάλογοι έχουν συμβολικά ονόματα ακριβώς όπως και τα αρχεία.

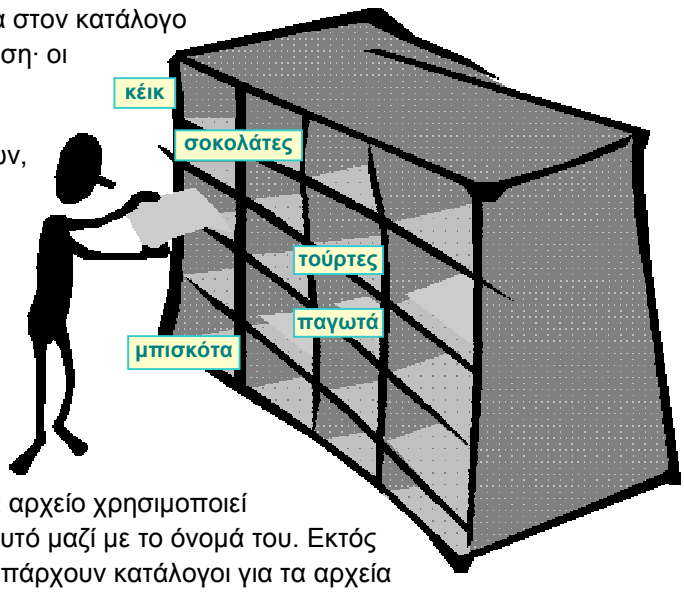
Η οργάνωση των αρχείων σε καταλόγους προσφέρει πολλά πλεονεκτήματα:

- Το σύστημα αρχείων δεν κρατά μια μεγάλη λίστα από όλα τα αρχεία, αλλά πολλές μικρές. Έτσι η αναζήτηση π.χ. ενός αρχείου είναι πιο γρήγορη, αν γνωρίζουμε τον κατάλογο στον οποίο αυτό βρίσκεται.
- Κάθε χρήστης του υπολογιστή μπορεί να κρατά τα αρχεία του σε διαφορετικό κατάλογο, στον οποίο μόνο αυτός έχει πρόσβαση. Έτσι, διευκολύνεται το έργο του συστήματος αρχείων ως προς τη διαχείριση των δικαιωμάτων πρόσβασης.

- Πολλά αρχεία μπορούν να έχουν το ίδιο όνομα, αρκεί να βρίσκονται σε διαφορετικό κατάλογο. Έτσι π.χ. κάθε χρήστης μπορεί να κρατά τα μηνύματα του ηλεκτρονικού ταχυδρομείου του σε ένα αρχείο με το όνομα «mailbox» χωρίς να ενδιαφέρεται αν και οι άλλοι χρήστες διαθέτουν τέτοιο αρχείο.
- Τα αρχεία του ΛΣ μπορούν να διατηρούνται χωριστά από τα αρχεία των χρηστών και μόνο οι *διαχειριστές του συστήματος* (system administrators) να έχουν πρόσβαση σ' αυτά.

Κατάλογοι ενός επιπέδου

Στην πιο απλή μορφή του συστήματος αρχείων, όπου ο αριθμός των χρηστών και των αρχείων δεν είναι μεγάλος, το σύστημα αρχείων κρατά έναν κεντρικό πίνακα με μια θέση για κάθε χρήστη. Κάθε θέση στον κεντρικό πίνακα αντιστοιχεί σε έναν κατάλογο. Όλα τα αρχεία του κάθε χρήστη βρίσκονται μέσα στον κατάλογο που του ανήκει, χωρίς καμία περαιτέρω οργάνωση· οι κατάλογοι δηλαδή περιέχουν μόνο αρχεία.



> Σε μια αντίστοιχη οργάνωση των συνταγών, κάθε χώρισμα στα ράφια της βιβλιοθήκης του ζαχαροπλάστη θα περιείχε μια γενική κατηγορία γλυκών, και σε κάθε χώρισμα οι συνταγές θα ήταν τοποθετημένες με τυχαίο τρόπο, χωρίς να έχουν χωριστεί παραπέρα σε φακέλους.

Στην οργάνωση αυτή, όλοι οι κατάλογοι βρίσκονται στην ίδια απόσταση από τον κεντρικό πίνακα. Για να αναφερθεί κανείς σε ένα αρχείο χρησιμοποιεί το όνομα του καταλόγου στον οποίο βρίσκεται αυτό μαζί με το όνομά του. Εκτός από τους καταλόγους των χρηστών μπορεί να υπάρχουν κατάλογοι για τα αρχεία του ΛΣ.

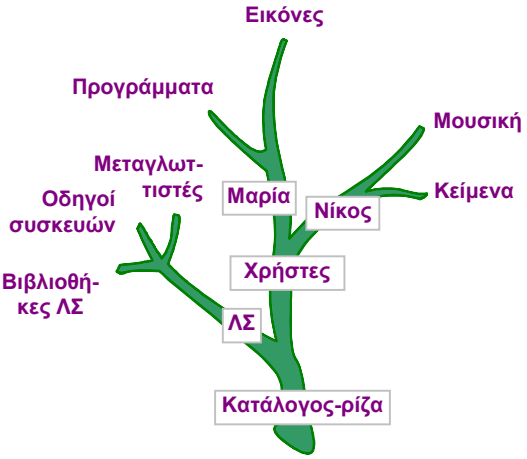
Κατάλογοι πολλαπλών επιπέδων

Όταν το σύστημα αρχείων οργανώνει τους καταλόγους σε πολλά επίπεδα, κάθε κατάλογος μπορεί να περιέχει άλλους καταλόγους, αλλά και αρχεία. Όταν ο κατάλογος Α περιέχει τον κατάλογο Β, ονομάζουμε τον Β *υποκατάλογο* (subdirectory) του Α και τον Α *κατάλογο-πατέρα* (parent directory) του Β. Έτσι δημιουργείται ένα «δένδρο» καταλόγων, όπου ο υποκατάλογος είναι «παρακλάδι» του καταλόγου που τον περιέχει και ο κεντρικός πίνακας καταλόγων του συστήματος αρχείων παίζει το ρόλο της ρίζας (root directory).

> Στην οργάνωση του συστήματος αρχείων που φαίνεται στο σχήμα της επόμενης σελίδας, ο κατάλογος-ρίζα περιέχει δυο βασικούς καταλόγους: ένα για τα αρχεία του ΛΣ και ένα για τα αρχεία των χρηστών. Ο κατάλογος για τα αρχεία των χρηστών περιέχει διάφορα αρχεία, και από έναν υποκατάλογο για κάθε χρήστη. Ο Νίκος π.χ. κρατά όλα τα αρχεία του στον κατάλογο που έχει το όνομά του. Για τα αρχεία όμως που περιέχουν μουσικά κομμάτια έχει φτιάξει ένα ξεχωριστό υποκατάλογο, με το όνομα «Μουσική»· το ίδιο έχει κάνει και για τα κείμενα που γράφει.

Για να προσδιορίσουμε ένα αρχείο σε ένα σύστημα καταλόγων με πολλά επίπεδα, πρέπει να δώσουμε μια σειρά από ονόματα καταλόγων, ξεκινώντας από τον κατάλογο-ρίζα, τα οποία προσδιορίζουν τον τελικό κατάλογο όπου βρίσκεται το

αρχείο. Αυτή η σειρά από ονόματα καταλόγων αποκαλείται *διαδρομή* ή *μονοπάτι* (path). Στα πιο πολλά ΛΣ μια διαδρομή δίνεται χωρίζοντας τα ονόματα των καταλόγων με κάποιο ειδικό χαρακτήρα, π.χ. «/» (όπως στο Unix) ή «\» (όπως στο DOS και τα Windows). Ο ειδικός χαρακτήρας αυτός μόνος του (ή στην αρχή μιας διαδρομής) συμβολίζει τον κατάλογο-ρίζα.



Η Μαρία έχει μια φωτογραφία στο αρχείο με όνομα «photo», μέσα στον κατάλογο με τις εικόνες. Η διαδρομή προς το αρχείο αυτό είναι: Κατάλογος-ρίζα → Χρήστες → Μαρία → Εικόνες → photo. Αν τα ονόματα των καταλογων χωρίζονται με το χαρακτήρα «\», η διαδρομή θα είναι «\Χρήστες\Μαρία\Εικόνες\photo»[1]. Ο μεταγλωττιστής της γλώσσας Pascal, που έχει το όνομα «pascal» θα κρατείται με τα αρχεία του ΛΣ, στον κατάλογο των μεταγλωττιστών. Η διαδρομή που προσδιορίζει το μεταγλωττιστή είναι: Κατάλογος-ρίζα → ΛΣ → Μεταγλωττιστές → pascal.

Οι διαδρομές αυτές που περιγράψαμε ορίζονται πάντα ως προς τον κατάλογο-ρίζα, είναι λοιπόν *απόλυτες διαδρομές* (absolute paths). Μπορεί όμως να θέλουμε να ορίσουμε μια διαδρομή ως προς έναν άλλο κατάλογο, να ορίσουμε δηλαδή μια *σχετική διαδρομή* (relative path).

Ο Νίκος θέλει να ορίσει τον κατάλογο που κρατά τα κείμενά του ως προς τον κατάλογο των χρηστών. Για το σκοπό αυτό θα ακολουθήσει το μονοπάτι του δένδρου από τον κατάλογο «Χρήστες» έως τον κατάλογο «Κείμενα», το οποίο είναι: Χρήστες → Νίκος → Κείμενα. Αν τα ονόματα των καταλόγων χωρίζονται με τον ειδικό χαρακτήρα «/», το σχετικό μονοπάτι θα είναι «Νίκος/Κείμενα». Αυτό προκύπτει εύκολα αν αφαιρέσουμε από την απόλυτη διαδρομή «/Χρήστες/Νίκος/Κείμενα» την απόλυτη διαδρομή /Χρήστες».

Είναι λοιπόν απλό να ορίσουμε μια σχετική διαδρομή που διασχίζει το δένδρο των καταλόγων από τη ρίζα προς τα «φύλλα». Τι κάνουμε όμως αν θέλουμε να διασχίσουμε το δένδρο από τα «φύλλα» προς τη ρίζα; Για το σκοπό αυτό, κάθε κατάλογος έχει ένα υποθετικό υποκατάλογο, με όνομα π.χ. «..», που παριστάνει τον κατάλογο-πατέρα του. Όταν σε μια σχετική διαδρομή εμφανίζεται αυτός ο υποκατάλογος, τότε η διαδρομή διασχίζει το δένδρο προς τη ρίζα. Αυτή η μέθοδος χρησιμοποιείται τόσο στα ΛΣ DOS/Windows όσο και στο Unix.

Ο κατάλογος μουσικής του Νίκου σε σχέση με τον κατάλογό του των κειμένων (Κείμενα → Νίκος → Μουσική) περιγράφεται από τη σχετική διαδρομή «..\Μουσική», ενώ η απόλυτη διαδρομή για αυτόν είναι η «\Χρήστες\Νίκος\Μουσική».

Αν θέλουμε να ορίσουμε το μεταγλωττιστή της Pascal ως προς τον κατάλογο της Μαρίας που περιέχει τα προγράμματά της, πρέπει να διατρέξουμε το δένδρο ως εξής: Προγράμματα → Μαρία → Χρήστες → Κατάλογος-ρίζα → ΛΣ → Μεταγλωττιστές. Κάθε μετάβαση προς τη ρίζα στη διαδρομή αυτή παριστάνεται με τον κατάλογο «..». Αν λοιπόν οι

[1] Τα ονόματα αρχείων και καταλόγων με ελληνικούς χαρακτήρες δεν επιτρέπονται σε όλα τα ΛΣ.

κατάλογοι χωρίζονται με το σύμβολο «\», η σχετική διαδρομή αυτή είναι «..\..\..\ΛΣ\Μεταγλωττιστές», και είναι ισοδύναμη με την απόλυτη διαδρομή «\ ΛΣ\Μεταγλωττιστές».

Λειτουργίες των καταλόγων

Οι κατάλογοι μπορούν να υλοποιηθούν σαν αρχεία ειδικού τύπου (π.χ. με ένα χαρακτηριστικό που προσδιορίζει ότι είναι κατάλογοι), τα οποία επεξεργάζεται το σύστημα αρχείων και περιέχουν πληροφορίες για τα αρχεία και τους άλλους καταλόγους που περιέχουν.

Οι βασικές λειτουργίες που μπορεί να εκτελέσει κανείς σε ένα κατάλογο είναι:

	Δημιουργία ενός νέου καταλόγου, ο οποίος περιέχεται σε ένα άλλο κατάλογο. Κατά τη δημιουργία του συστήματος αρχείων, ο κατάλογος-ρίζα προϋπάρχει, οπότε αρχίζουν και δημιουργούνται κατάλογοι μέσα σε αυτόν. Αρχικά ο νέος κατάλογος είναι κενός, δηλαδή δεν περιέχει αρχεία ή άλλους καταλόγους.
	Διαγραφή ενός καταλόγου· για να μπορεί να διαγραφεί ένας κατάλογος πρέπει να είναι κενός, να μην περιέχει δηλαδή αρχεία ή άλλους καταλόγους. Ο κατάλογος-ρίζα δεν μπορεί να διαγραφεί.
	Ανάγνωση των περιεχομένων ενός καταλόγου, δηλαδή απαρίθμηση των αρχείων και των υποκαταλόγων που αυτός περιέχει. Η ανάγνωση γίνεται συνήθως από ειδικά προγράμματα που εμφανίζουν στο χρήστη τα περιεχόμενα του καταλόγου.
	Εγγραφή σε ένα κατάλογο· η λειτουργία αυτή γίνεται μόνο έμμεσα, όταν δημιουργείται ένα νέο αρχείο ή υποκατάλογος μέσα στον κατάλογο, όταν διαγράφεται ένα αρχείο ή υποκατάλογος ή όταν μετονομάζεται κάποιο από τα περιεχόμενα του καταλόγου.
	Μετονομασία του καταλόγου· το νέο όνομα δεν πρέπει να υπάρχει ήδη στον κατάλογο-πατέρα. Φυσικά, το όνομα του καταλόγου-ρίζα δεν μπορεί να αλλάξει.

Ανακεφαλαίωση

Το Σύστημα Αρχείων είναι ένα πολύ σημαντικό τμήμα ενός ΛΣ που έχει ως σκοπό να οργανώσει τα δεδομένα που αποθηκεύονται στη δευτερεύουσα μνήμη. Τα δεδομένα είναι μοιρασμένα σε αρχεία, κάθε ένα από τα οποία έχει ένα όνομα. Οι διεργασίες αναφέρονται στο αρχείο με το όνομά του, και μπορούν να εκτελέσουν σε αυτό διάφορες λειτουργίες. Οι πιο σημαντικές είναι η ανάγνωση και η εγγραφή δεδομένων, τα οποία είναι μόνιμα, δηλαδή δε χάνονται όταν ο υπολογιστής σταματήσει να λειτουργεί. Η οργάνωση των αρχείων γίνεται καλύτερα με το μοίρασμά τους σε καταλόγους οι οποίοι περιέχουν ομοειδή αρχεία. Οι κατάλογοι μπορούν να είναι οργανωμένοι σε ένα ή σε πολλά επίπεδα· στη δεύτερη περίπτωση οι κατάλογοι δημιουργούν μια δομή δένδρου. Οι αναφορές στα αρχεία γίνονται μέσω διαδρομών, ακολουθιών δηλαδή από καταλόγους.

Γλωσσάριο όρων

Απόλυτη Διαδρομή	*Absolute Path*
Αρχείο	*File*
Αρχείο Μόνο για Ανάγνωση	*Read Only File*
Διαδρομή ή Μονοπάτι	*Path*
Διαχειριστής Συστήματος	*System Administrator*
Δικαιώματα Προσπέλασης	*Access Rights*
Κατάλογος ή Ευρετήριο	*Directory*
Κατάλογος-Πατέρας	*Parent Directory*
Κατάλογος-Ρίζα	*Root Directory*
Όνομα Αρχείου	*File Name*
Προέκταση Ονόματος	*File Name Extension*
Σύστημα Αρχείων	*File System*
Σχετική Διαδρομή	*Relative Path*
Τμήμα Ελέγχου Αρχείου - ΤΕΑ	*File Control Block - FCB*
Υποκατάλογος	*Subdirectory*
Χαρακτηριστικό	*Attribute - Property*

Ερωτήσεις

? Σε τι χρησιμεύουν τα ονόματα των αρχείων;

? Ποιες λειτουργίες μπορούμε να εκτελέσουμε σε ένα αρχείο;

? Σε τι διαφέρει μία απόλυτη από μία σχετική διαδρομή;

? Ποιες λειτουργίες μπορούμε να εκτελέσουμε σε έναν κατάλογο; Είναι οι ίδιες με αυτές των αρχείων και γιατί;

Τι μάθαμε σε αυτό το κεφάλαιο

- Το Σύστημα Αρχείων είναι ένα πολύ σημαντικό τμήμα ενός ΛΣ που έχει ως σκοπό να οργανώσει τα δεδομένα που αποθηκεύονται στη δευτερεύουσα μνήμη.
- Τα δεδομένα είναι μοιρασμένα σε αρχεία, κάθε ένα από τα οποία έχει ένα όνομα.
- Οι διεργασίες αναφέρονται στο αρχείο με το όνομά του, και μπορούν να εκτελέσουν σε αυτό διάφορες λειτουργίες.
- Η οργάνωση των αρχείων γίνεται καλύτερα με το μοίρασμά τους σε καταλόγους οι οποίοι περιέχουν ομοειδή αρχεία.
- Οι αναφορές στα αρχεία γίνονται μέσω διαδρομών, ακολουθιών δηλαδή από καταλόγους.

Βιβλιογραφία – Πηγές

- Παπακωσταντίνου Γ., Ν. Μπιλάλη, Π. Τσανάκα, Λειτουργικά Συστήματα: Αρχές Λειτουργίας, Συμμετρία, 1989.
- Silberschatz A., Peterson J., Galvin P., Operating System Concepts, Addison - Wesley, 1991.